GCC Trees
Intermediate Representation

정원교
weongyo@hotmail.com

2003년 11월 3일

목 차

제 1 절 14 주째 강의를 시작하며

이번 주에는 GCC 내에서 C 와 C++ 를 표현하는데 사용되는 internal representation 을 설명하도록 하겠습니다. 하지만 C++ 에서 사용되는 TREE node 들에 대한 설명은 전혀 하지 않습니다. 중간 부분에 나오는 TREE node 들의 예제들 중 본 문서에 나와 있지 않는 node 들의 예제의 경우 제 메일로 보내주시면 감사하겠습니다.

제 2 절 tree struct

$prefix/gcc/config.h - tree 의 선언

```
union tree_node;
typedef union tree_node *tree;
```

tree node 는 date type 과 variable, expression, statemet 들을 표현할 수 있습니다. 각 node 는 그것이 나타내는 것이 어떤 종류인지를 말해 주는 TREE_CODE 를 가지고 있으며 몇몇 공통된 code 들은 아래와 같은 것이 있습니다.

INTEGER_TYPE – 정수형 type 을 나타냅니다.
ARRAY_TYPE – 포인터형 type 을 나타냅니다.
VAR_DECL – declared variable 을 나타냅니다.
INTEGER_CST – constant integer value 를 나타냅니다.
PLUS_EXPR – 합계 (표현식) 을 나타냅니다.

tree node 의 내용으로써 모든 Node 들이 공유하는 몇몇 field 들이 있습니다. 각 TREE_CODE 는 또한 특정-목적을 위한 여러 field 들도 가지고 있습니다. node 의 field 들은 절대 직접접근을 하지 않고 그것을 연결해 주는 연결자 매크로를 통해서 접근해야 합니다.

2.1 union tree_node

tree node 의 전체 내용들을 정의한다. 이것은 아래에 선언될 여러 node 의 type 들 중에서 하나를 가질 것이다.

$prefix/gcc/tree.h - union tree_node 공용체

```
union tree_node {
  struct tree_common common;
  struct tree_int_cst int_cst;
  struct tree_real_cst real_cst;
  struct tree_vector vector;
```

```
  struct tree_string string;
  struct tree_complex complex;
  struct tree_identifier identifier;
  struct tree_decl decl;
  struct tree_type type;
  struct tree_list list;
  struct tree_vec vec;
  struct tree_exp exp;
  struct tree_block block;
 };
```

2.2 struct tree_common

모든 종류의 tree node 들은 이 구조체로 시작한다. 그래서 모든 node 들은 위의 field 들을 모두 공통적으로 가지고 있다.

$prefix/gcc/tree.h - struct tree_common 구조체

```
struct tree_common {
  tree chain;
  tree type;

  ENUM_BITFIELD(tree_code) code : 8;

  unsigned side_effects_flag : 1;
  unsigned constant_flag : 1;
  unsigned addressable_flag : 1;
  unsigned volatile_flag : 1;
  unsigned readonly_flag : 1;
  unsigned unsigned_flag : 1;
  unsigned asm_written_flag: 1;
  unsigned unused_0 : 1;

  unsigned used_flag : 1;
  unsigned nothrow_flag : 1;
  unsigned static_flag : 1;
  unsigned public_flag : 1;
  unsigned private_flag : 1;
  unsigned protected_flag : 1;
  unsigned bounded_flag : 1;
  unsigned deprecated_flag : 1;

  unsigned lang_flag_0 : 1;
  unsigned lang_flag_1 : 1;
  unsigned lang_flag_2 : 1;
  unsigned lang_flag_3 : 1;
  unsigned lang_flag_4 : 1;
  unsigned lang_flag_5 : 1;
  unsigned lang_flag_6 : 1;
  unsigned unused_1 : 1;
};
```

다음의 table 은 위의 각 flag 들의 사용에 관한 내용이며 그들이 어떤 type 에서 정의될 수 있는가에 대한 설명을 하고 있습니다. expression 들은 decl 들을 포함하고 있음을 아시기 바랍니다.

addressable_flag:

    TREE_ADDRESSABLE 사용
        VAR_DECL, FUNCTION_DECL, FIELD_DECL, CONSTRUCTOR, LABEL_DECL,
        ..._TYPE, IDENTIFIER_NODE.
        STMT_EXPR 내에서는 우리가 enclosed expression 의 결과를
        원하는 것을 의미합니다.

static_flag:

    TREE_STATIC 사용
        VAR_DECL, FUNCTION_DECL, CONSTRUCTOR, ADDR_EXPR
    TREE_NO_UNUSED_WARNING 사용
        CONVERT_EXPR, NOP_EXPR, COMPOUND_EXPR
    TREE_VIA_VIRTUAL 사용
        TREE_LIST 혹은 TREE_VEC
    TREE_CONSTANT_OVERFLOW 사용
        INTEGER_CST, REAL_CST, COMPLEX_CST, VECTOR_CST
    TREE_SYMBOL_REFERENCED 사용
        IDENTIFIER_NODE
    CLEANUP_EH_ONLY 사용
        Block 의 cleanup list 의 TARGET_EXPR,
        WITH_CLEANUP_EXPR, CLEANUP_STMT, TREE_LIST element 들.

public_flag:

    TREE_OVERFLOW 사용
        INTEGER_CST, REAL_CST, COMPLEX_CST, VECTOR_CST
    TREE_PUBLIC 사용
        VAR_DECL 혹은 FUNCTION_DECL 혹은 IDENTIFIER_NODE
    TREE_VIA_PUBLIC 사용
        TREE_LIST 혹은 TREE_VEC
    EXPR_WFL_EMIT_LINE_NOTE 사용
        EXPR_WITH_FILE_LOCATION

private_flag:

    TREE_VIA_PRIVATE 사용
        TREE_LIST 혹은 TREE_VEC
    TREE_PRIVATE 사용
        ..._DECL

protected_flag:

    TREE_VIA_PROTECTED 사용
        TREE_LIST
        TREE_VEC
    TREE_PROTECTED 사용
        BLOCK
        ..._DECL

side_effects_flag:

```
    TREE_SIDE_EFFECTS 사용
        모든 표현식

volatile_flag:

    TREE_THIS_VOLATILE 사용
        모든 표현식
    TYPE_VOLATILE 사용
        ..._TYPE

readonly_flag:

    TREE_READONLY 사용
        모든 표현식
    TYPE_READONLY 사용
        ..._TYPE

constant_flag:

    TREE_CONSTANT 사용
        모든 표현식

unsigned_flag:

    TREE_UNSIGNED 사용
        INTEGER_TYPE, ENUMERAL_TYPE, FIELD_DECL
    DECL_BUILT_IN_NONANSI 사용
        FUNCTION_DECL
    SAVE_EXPR_NOPLACEHOLDER 사용
        SAVE_EXPR

asm_written_flag:

    TREE_ASM_WRITTEN 사용
        VAR_DECL, FUNCTION_DECL, RECORD_TYPE, UNION_TYPE, QUAL_UNION_TYPE
        BLOCK

used_flag:

    TREE_USED 사용
        표현식들, IDENTIFIER_NODE

nothrow_flag:

    TREE_NOTHROW 사용
        CALL_EXPR, FUNCTION_DECL

bounded_flag:

    TREE_BOUNDED 사용
        표현식, VAR_DECL, PARM_DECL, FIELD_DECL, FUNCTION_DECL,
```

```
          IDENTIFIER_NODE
      TYPE_BOUNDED 사용
          ..._TYPE

  deprecated_flag:

       TREE_DEPRECATED 사용
          ..._DECL
```

2.3 struct tree_int_cst

$prefix/gcc/tree.h - struct tree_int_cst
상수들을 표현하기 위한 구조체 TREE 이다.

```
struct tree_int_cst {
  struct tree_common common;
  rtx rtl;
  struct {
    unsigned HOST_WIDE_INT low;
    HOST_WIDE_INT high;
  } int_cst;
};
```

구성요소에 대한 설명은 다음과 같다.

rtl

register transfer language (rtl) info 에 연결된 것 처럼 행동한다.

int_cst

여기에 하위-구조체가 필요한 이유는 함수 'const_hash' 가 하나의 unit 으로 두 word 들을 훑기를 원하고 하위-구조체의 주소를 가질 수 있음으로써 적당한 충괄적인 bounded pointer 를 산출할 수 있다.

2.4 struct tree_real_cst

$prefix/gcc/tree.h - struct tree_real_cst
REAL_CST node 에서 사용되며, 'double' 혹은 'long' 들의 배열로 실수값을 나타낸다.

```
struct tree_real_cst {
  struct tree_common common;
  rtx rtl;
  REAL_VALUE_TYPE real_cst;
};
```

구성요소에 대한 설명은 다음과 같다.

rtl

register transfer language (rtl) info 에 연결된 것 처럼 행동한다.

2.5 struct tree_vector

$prefix/gcc/tree.h - struct tree_vector
VECTOR_CST node 를 위해 사용된다.

```
struct tree_vector {
  struct tree_common common;
  rtx rtl;
  tree elements;
};
```

2.6 struct tree_string

$prefix/gcc/tree.h - struct tree_string
STRING_CST 를 위해서 사용되는 구조체이다.

```
struct tree_string {
  struct tree_common common;
  rtx rtl;
  int length;
  const char *pointer;
};
```

구성요소에 대한 설명은 다음과 같다.

rtl

register transfer language (rtl) info 에 연결된 것 처럼 행동한다.

2.7 struct tree_complex

$prefix/gcc/tree.h - struct tree_complex
COMPLEX_CST node 를 위해서 사용된다.

```
struct tree_complex {
  struct tree_common common;
  rtx rtl;
  tree real;
  tree imag;
};
```

구성요소에 대한 설명은 다음과 같다.

rtl

register transfer language (rtl) info 에 연결된 것 처럼 행동한다.

real

실수 표현식을 위한 TREE node 를 가진다.

imag

허수 표현식을 위한 TREE node 를 가진다.

2.8 struct tree_identifier

$prefix/gcc/tree.h - struct tree_identifier
Identifier (식별자) 를 위한 구조체이다. 하지만 이를 사용해서만 Identifier 를 나타내는 것이 아닌데, C 언어에서는 대부분 struct lang_identifier 구조체를 사용함으로써 표현되며, 이를 위한 설정은 $prefix/gcc/toplev.c 의 초반 설정부분에서 볼 수 있다.

```
struct tree_identifier {
  struct tree_common common;
  struct ht_identifier id;
};
```

2.9 struct tree_decl

$prefix/gcc/tree.h - struct tree_decl 구조체는 선언된 name 들과 정보를 나타낸다.

```
struct tree_decl {
  struct tree_common common;
  const char *filename;
  int linenum;
  unsigned int uid;
  tree size;
  ENUM_BITFIELD(machine_mode) mode : 8;

  unsigned external_flag : 1;
  unsigned nonlocal_flag : 1;
  unsigned regdecl_flag : 1;
  unsigned inline_flag : 1;
  unsigned bit_field_flag : 1;
  unsigned virtual_flag : 1;
  unsigned ignored_flag : 1;
  unsigned abstract_flag : 1;

  unsigned in_system_header_flag : 1;
  unsigned common_flag : 1;
  unsigned defer_output : 1;
  unsigned transparent_union : 1;
  unsigned static_ctor_flag : 1;
  unsigned static_dtor_flag : 1;
  unsigned artificial_flag : 1;
  unsigned weak_flag : 1;

  unsigned non_addr_const_p : 1;
  unsigned no_instrument_function_entry_exit : 1;
  unsigned comdat_flag : 1;
  unsigned malloc_flag : 1;
  unsigned no_limit_stack : 1;
  ENUM_BITFIELD(built_in_class) built_in_class : 2;
  unsigned pure_flag : 1;

  unsigned pointer_depth : 2;
  unsigned non_addressable : 1;
  unsigned user_align : 1;
  unsigned uninlinable : 1;
```

```
  unsigned lang_flag_0 : 1;
  unsigned lang_flag_1 : 1;
  unsigned lang_flag_2 : 1;
  unsigned lang_flag_3 : 1;
  unsigned lang_flag_4 : 1;
  unsigned lang_flag_5 : 1;
  unsigned lang_flag_6 : 1;
  unsigned lang_flag_7 : 1;

  union {
    enum built_in_function f;
    HOST_WIDE_INT i;
    struct {
      unsigned int align : 24;
      unsigned int off_align : 8;
    } a;
  } u1;

  tree size_unit;
  tree name;
  tree context;
  tree arguments;
  tree result;
  tree initial;
  tree abstract_origin;
  tree assembler_name;
  tree section_name;
  tree attributes;
  rtx rtl;
  rtx live_range_rtl;

  union {
    struct function *f;
    rtx r;
    tree t;
    int i;
  } u2;

  tree saved_tree;

  tree inlined_fns;

  tree vindex;
  HOST_WIDE_INT pointer_alias_set;

  struct lang_decl *lang_specific;
};
```

구성요소에 대한 설명은 다음과 같다.

u1.f

DECL_BUILT_IN 가 FUNCTION_DECL 를 위해 어떤 것을 잡고 있는 상황에서는 이것은 DECL_FUNCTION_CODE 이다.

u1.i

DECL_BUILT_IN 가 FUNCTION_DECL 를 위해 어던 것을 잡고 있지 않은 상황에서는 이것은 언어-독립적인 code 로 사용된다.

u1.a

DECL_ALIGN 와 DECL_OFFSET_ALIGN. (이것들은 FUNCTION_DECL 들을 위해 사용되지 않는다.)

arguments

DECL_FIELD_OFFSET 용으로 또한 사용된다.

result

DECL_BIT_FIELD_TYPE 용으로 또한 사용된다.

initial

DECL_QUALIFIER 용으로 또한 사용된다

rtl

Object 를 위한 RTL representation.

u2

FUNCTION_DECL 에서 만약 그것이 inline 이라면 saved insn chain 을 가지고 있다.
FIELD_DECL 에서는 DECL_FIELD_BIT_OFFSET 이다.
PARM_DECL 에서는 data 가 실제로 통과되는 register 의 stack slot 을 위한 RTL 을 가지고 있다.
LABEL_DECL 부분에서는 Chill 과 Java 를 위해, VAR_DECL 에서는 C++ 과 Java 를 위해 사용된다.

saved_tree

FUNCTION_DECL 에서 이것은 DECL_SAVED_TREE 이다.

inlined_fns

FUNCTION_DECL 에서 이것들은 FUNCTION_DECL 가 유지되는 동안에 가능한 유지되기를 바라는 function data 이다.

lang_specific

세부사항이 사용하는 언어에 따라 의존하는 구조체를 가르킨다.

2.10 struct tree_type

$prefix/gcc/tree.h - struct tree_type 구조체는 data type 들을 표현하기 위한 것이다.

```
struct tree_type {
  struct tree_common common;
  tree values;
  tree size;
  tree size_unit;
```

```
  tree attributes;
  unsigned int uid;

  unsigned int precision : 9;
  ENUM_BITFIELD(machine_mode) mode : 7;

  unsigned string_flag : 1;
  unsigned no_force_blk_flag : 1;
  unsigned needs_constructing_flag : 1;
  unsigned transparent_union_flag : 1;
  unsigned packed_flag : 1;
  unsigned restrict_flag : 1;
  unsigned pointer_depth : 2;

  unsigned lang_flag_0 : 1;
  unsigned lang_flag_1 : 1;
  unsigned lang_flag_2 : 1;
  unsigned lang_flag_3 : 1;
  unsigned lang_flag_4 : 1;
  unsigned lang_flag_5 : 1;
  unsigned lang_flag_6 : 1;
  unsigned user_align : 1;

  unsigned int align;
  tree pointer_to;
  tree reference_to;
  union {int address; char *pointer; } symtab;
  tree name;
  tree minval;
  tree maxval;
  tree next_variant;
  tree main_variant;
  tree binfo;
  tree context;
  HOST_WIDE_INT alias_set;
  struct lang_type *lang_specific;
};
```

구성요소에 대한 설명은 다음과 같다.

lang_specific

세부사항이 사용하는 언어에 따라 의존하는 구조체를 가르킨다.

2.11 struct tree_list

$prefix/gcc/tree.h - struct tree_list

```
struct tree_list {
  struct tree_common common;
  tree purpose;
  tree value;
};
```

2.12 struct tree_vec

$prefix/gcc/tree.h - struct tree_vec

```
struct tree_vec {
  struct tree_common common;
  int length;
  tree a[1];
};
```

2.13 struct tree_exp

$prefix/gcc/tree.h - struct tree_exp 구조체는 expression 을 표현하기 위한 것이다.

```
struct tree_exp {
  struct tree_common common;
  int complexity;
  tree operands[1];
};
```

2.14 struct tree_block

$prefix/gcc/tree.h - struct tree_block 구조체는 BLOCK node 에서 사용되는 것이다.

```
struct tree_block {
  struct tree_common common;

  unsigned handler_block_flag : 1;
  unsigned abstract_flag : 1;
  unsigned block_num : 30;

  tree vars;
  tree subblocks;
  tree supercontext;
  tree abstract_origin;
  tree fragment_origin;
  tree fragment_chain;
};
```

제 3 절 tree macro

TREE 를 위한 구조체에 접근하기 위한 매크로를 이 절에서는 언급합니다. 접근 매크로로는 tree 관련 전역 변수나 구조체, 함수 등을 접근하는 것이 있고, 각 tree 가 가지는 속성에 따라 접근하는 것들이 있다.

3.1 tree 용 전역변수에 접근하기 위한 매크로

```
#define NUM_TREE_CODES ((int) LAST_AND_UNUSED_TREE_CODE)
```

Language-independent tree code 들의 갯수.

```
#define TREE_CODE_CLASS(CODE)   tree_code_type[(int) (CODE)]
```

enum tree_code 의 순서로 나열되어 있습니다. 이 변수는 한 문자를 포함하고 있습니다.

비교 expression 을 위한 '¡', unary arithmetic expression 을 위한 '1', binary arithmetic expression 을 위한 '2', 다른 expression 들의 type 들을 위한 'e', reference 을 위한 'r', constant 를 위한 'c', decl 를 위한 'd', type 을 위한 't', statement 를 위한 's', 그리고 다른 모든 것 (TREE_LIST, IDENTIFIER 등등) 을 위한 'x' 가 있습니다.

```
#define IS_EXPR_CODE_CLASS(CLASS) \
  ((CLASS) == '<' || (CLASS) == '1' || (CLASS) == '2' || (CLASS) == 'e')
```

CLASS 가 expression 에 관한 tree-code class 일 경우 0 이 아닌 값을 되돌려줍니다.

```
#define TREE_CODE_LENGTH(CODE)  tree_code_length[(int) (CODE)]
```

각각의 tree-node 종류에서의 argument-word 들의 갯수(number).

3.2 tree 용 구조체에 접근하기 위한 매크로

모든 tree node 들이 가지고 있는 field 들에 관한 연결자 매크로를 정의합니다. (비록 몇몇 field 들은 모든 경우의 node 들에게 사용되는건 아니지만).

3.2.1 tree_common 구조체

```
#define TREE_CODE(NODE) ((enum tree_code) (NODE)->common.code)
#define TREE_SET_CODE(NODE, VALUE) \
((NODE)->common.code = (ENUM_BITFIELD (tree_code)) (VALUE))
```

tree-code 는 이것이 어떤 종류의 node 인지를 말해줍니다. Code 들은 tree.def 내에 정의되어 있습니다.

```
#if defined ENABLE_TREE_CHECKING && (GCC_VERSION >= 2007)
```

검사기능이 enable 일 경우 tree node 가 잘못된 형태로 접근될 경우 오류를 발생할 것입니다. 매크로는 fatal error 로 끝마칩니다. 만약 이 정의가 참일 경우와 아래와 같은 것이 정의가 됩니다.

```
#define TREE_CHECK(t, code) __extension__                   \
({  const tree __t = (t);                                   \
    if (TREE_CODE(__t) != (code))                           \
      tree_check_failed (__t, code, __FILE__, __LINE__, __FUNCTION__);  \
    __t; })
#define TREE_CLASS_CHECK(t, class) __extension__            \
({  const tree __t = (t);                                   \
    if (TREE_CODE_CLASS(TREE_CODE(__t)) != (class))         \
      tree_class_check_failed (__t, class, __FILE__, __LINE__,      \
                  __FUNCTION__);                            \
    __t; })
```

TREE_CODE 를 검사할 때 사용됩니다.

```
#define CST_OR_CONSTRUCTOR_CHECK(t) __extension__           \
({  const tree __t = (t);                                   \
    enum tree_code const __c = TREE_CODE(__t);              \
    if (__c != CONSTRUCTOR && TREE_CODE_CLASS(__c) != 'c')      \
      tree_check_failed (__t, CONSTRUCTOR, __FILE__, __LINE__,      \
```

```
              __FUNCTION__);                                    \
    __t; })
#define EXPR_CHECK(t) __extension__                             \
({  const tree __t = (t);                                       \
    char const __c = TREE_CODE_CLASS(TREE_CODE(__t));           \
    if (__c != 'r' && __c != 's' && __c != '<'                  \
    && __c != '1' && __c != '2' && __c != 'e')                  \
      tree_class_check_failed (__t, 'e', __FILE__, __LINE__,    \
                  __FUNCTION__);                                \
    __t; })
```

이 검사들은 특별한 경우에만 수행되어야 한다.

하지만 ENABLE_TREE_CHECKING 가 아니고 혹은 gcc 가 아닐 경우는 아래와 같이 선언된다.

```
#define TREE_CHECK(t, code)     (t)
#define TREE_CLASS_CHECK(t, code)   (t)
#define CST_OR_CONSTRUCTOR_CHECK(t) (t)
#define EXPR_CHECK(t)           (t)
```

TREE_CODE 를 검사하지 않는다.

```
#define ERROR_MARK_CHECK(t)             TREE_CHECK (t, ERROR_MARK)
#define IDENTIFIER_NODE_CHECK(t)        TREE_CHECK (t, IDENTIFIER_NODE)
#define TREE_LIST_CHECK(t)              TREE_CHECK (t, TREE_LIST)
#define TREE_VEC_CHECK(t)               TREE_CHECK (t, TREE_VEC)
#define BLOCK_CHECK(t)                  TREE_CHECK (t, BLOCK)
#define VOID_TYPE_CHECK(t)              TREE_CHECK (t, VOID_TYPE)
#define INTEGER_TYPE_CHECK(t)           TREE_CHECK (t, INTEGER_TYPE)
#define REAL_TYPE_CHECK(t)              TREE_CHECK (t, REAL_TYPE)
#define COMPLEX_TYPE_CHECK(t)           TREE_CHECK (t, COMPLEX_TYPE)
#define VECTOR_TYPE_CHECK(t)            TREE_CHECK (t, VECTOR_TYPE)
#define ENUMERAL_TYPE_CHECK(t)          TREE_CHECK (t, ENUMERAL_TYPE)
#define BOOLEAN_TYPE_CHECK(t)           TREE_CHECK (t, BOOLEAN_TYPE)
#define CHAR_TYPE_CHECK(t)              TREE_CHECK (t, CHAR_TYPE)
#define POINTER_TYPE_CHECK(t)           TREE_CHECK (t, POINTER_TYPE)
#define OFFSET_TYPE_CHECK(t)            TREE_CHECK (t, OFFSET_TYPE)
#define REFERENCE_TYPE_CHECK(t)         TREE_CHECK (t, REFERENCE_TYPE)
#define METHOD_TYPE_CHECK(t)            TREE_CHECK (t, METHOD_TYPE)
#define FILE_TYPE_CHECK(t)              TREE_CHECK (t, FILE_TYPE)
#define ARRAY_TYPE_CHECK(t)             TREE_CHECK (t, ARRAY_TYPE)
#define SET_TYPE_CHECK(t)               TREE_CHECK (t, SET_TYPE)
#define RECORD_TYPE_CHECK(t)            TREE_CHECK (t, RECORD_TYPE)
#define UNION_TYPE_CHECK(t)             TREE_CHECK (t, UNION_TYPE)
#define QUAL_UNION_TYPE_CHECK(t)        TREE_CHECK (t, QUAL_UNION_TYPE)
#define FUNCTION_TYPE_CHECK(t)          TREE_CHECK (t, FUNCTION_TYPE)
#define LANG_TYPE_CHECK(t)              TREE_CHECK (t, LANG_TYPE)
#define INTEGER_CST_CHECK(t)            TREE_CHECK (t, INTEGER_CST)
#define REAL_CST_CHECK(t)               TREE_CHECK (t, REAL_CST)
#define COMPLEX_CST_CHECK(t)            TREE_CHECK (t, COMPLEX_CST)
#define VECTOR_CST_CHECK(t)             TREE_CHECK (t, VECTOR_CST)
#define STRING_CST_CHECK(t)             TREE_CHECK (t, STRING_CST)
```

```
#define FUNCTION_DECL_CHECK(t)          TREE_CHECK (t, FUNCTION_DECL)
#define LABEL_DECL_CHECK(t)             TREE_CHECK (t, LABEL_DECL)
#define CONST_DECL_CHECK(t)             TREE_CHECK (t, CONST_DECL)
#define TYPE_DECL_CHECK(t)              TREE_CHECK (t, TYPE_DECL)
#define VAR_DECL_CHECK(t)               TREE_CHECK (t, VAR_DECL)
#define PARM_DECL_CHECK(t)              TREE_CHECK (t, PARM_DECL)
#define RESULT_DECL_CHECK(t)            TREE_CHECK (t, RESULT_DECL)
#define FIELD_DECL_CHECK(t)             TREE_CHECK (t, FIELD_DECL)
#define NAMESPACE_DECL_CHECK(t)         TREE_CHECK (t, NAMESPACE_DECL)
#define COMPONENT_REF_CHECK(t)          TREE_CHECK (t, COMPONENT_REF)
#define BIT_FIELD_REF_CHECK(t)          TREE_CHECK (t, BIT_FIELD_REF)
#define INDIRECT_REF_CHECK(t)           TREE_CHECK (t, INDIRECT_REF)
#define BUFFER_REF_CHECK(t)             TREE_CHECK (t, BUFFER_REF)
#define ARRAY_REF_CHECK(t)              TREE_CHECK (t, ARRAY_REF)
#define ARRAY_RANGE_REF_CHECK(t)        TREE_CHECK (t, ARRAY_RANGE_REF)
#define VTABLE_REF_CHECK(t)             TREE_CHECK (t, VTABLE_REF)
#define CONSTRUCTOR_CHECK(t)            TREE_CHECK (t, CONSTRUCTOR)
#define COMPOUND_EXPR_CHECK(t)          TREE_CHECK (t, COMPOUND_EXPR)
#define MODIFY_EXPR_CHECK(t)            TREE_CHECK (t, MODIFY_EXPR)
#define INIT_EXPR_CHECK(t)              TREE_CHECK (t, INIT_EXPR)
#define TARGET_EXPR_CHECK(t)            TREE_CHECK (t, TARGET_EXPR)
#define COND_EXPR_CHECK(t)              TREE_CHECK (t, COND_EXPR)
#define BIND_EXPR_CHECK(t)              TREE_CHECK (t, BIND_EXPR)
#define CALL_EXPR_CHECK(t)              TREE_CHECK (t, CALL_EXPR)
#define METHOD_CALL_EXPR_CHECK(t)       TREE_CHECK (t, METHOD_CALL_EXPR)
#define WITH_CLEANUP_EXPR_CHECK(t)      TREE_CHECK (t, WITH_CLEANUP_EXPR)
#define CLEANUP_POINT_EXPR_CHECK(t)     TREE_CHECK (t, CLEANUP_POINT_EXPR)
#define PLACEHOLDER_EXPR_CHECK(t)       TREE_CHECK (t, PLACEHOLDER_EXPR)
#define WITH_RECORD_EXPR_CHECK(t)       TREE_CHECK (t, WITH_RECORD_EXPR)
#define PLUS_EXPR_CHECK(t)              TREE_CHECK (t, PLUS_EXPR)
#define MINUS_EXPR_CHECK(t)             TREE_CHECK (t, MINUS_EXPR)
#define MULT_EXPR_CHECK(t)              TREE_CHECK (t, MULT_EXPR)
#define TRUNC_DIV_EXPR_CHECK(t)         TREE_CHECK (t, TRUNC_DIV_EXPR)
#define CEIL_DIV_EXPR_CHECK(t)          TREE_CHECK (t, CEIL_DIV_EXPR)
#define FLOOR_DIV_EXPR_CHECK(t)         TREE_CHECK (t, FLOOR_DIV_EXPR)
#define ROUND_DIV_EXPR_CHECK(t)         TREE_CHECK (t, ROUND_DIV_EXPR)
#define TRUNC_MOD_EXPR_CHECK(t)         TREE_CHECK (t, TRUNC_MOD_EXPR)
#define CEIL_MOD_EXPR_CHECK(t)          TREE_CHECK (t, CEIL_MOD_EXPR)
#define FLOOR_MOD_EXPR_CHECK(t)         TREE_CHECK (t, FLOOR_MOD_EXPR)
#define ROUND_MOD_EXPR_CHECK(t)         TREE_CHECK (t, ROUND_MOD_EXPR)
#define RDIV_EXPR_CHECK(t)              TREE_CHECK (t, RDIV_EXPR)
#define EXACT_DIV_EXPR_CHECK(t)         TREE_CHECK (t, EXACT_DIV_EXPR)
#define FIX_TRUNC_EXPR_CHECK(t)         TREE_CHECK (t, FIX_TRUNC_EXPR)
#define FIX_CEIL_EXPR_CHECK(t)          TREE_CHECK (t, FIX_CEIL_EXPR)
#define FIX_FLOOR_EXPR_CHECK(t)         TREE_CHECK (t, FIX_FLOOR_EXPR)
#define FIX_ROUND_EXPR_CHECK(t)         TREE_CHECK (t, FIX_ROUND_EXPR)
#define FLOAT_EXPR_CHECK(t)             TREE_CHECK (t, FLOAT_EXPR)
#define NEGATE_EXPR_CHECK(t)            TREE_CHECK (t, NEGATE_EXPR)
#define MIN_EXPR_CHECK(t)               TREE_CHECK (t, MIN_EXPR)
#define MAX_EXPR_CHECK(t)               TREE_CHECK (t, MAX_EXPR)
#define ABS_EXPR_CHECK(t)               TREE_CHECK (t, ABS_EXPR)
#define FFS_EXPR_CHECK(t)               TREE_CHECK (t, FFS_EXPR)
```

```
#define LSHIFT_EXPR_CHECK(t)            TREE_CHECK (t, LSHIFT_EXPR)
#define RSHIFT_EXPR_CHECK(t)            TREE_CHECK (t, RSHIFT_EXPR)
#define LROTATE_EXPR_CHECK(t)           TREE_CHECK (t, LROTATE_EXPR)
#define RROTATE_EXPR_CHECK(t)           TREE_CHECK (t, RROTATE_EXPR)
#define BIT_IOR_EXPR_CHECK(t)           TREE_CHECK (t, BIT_IOR_EXPR)
#define BIT_XOR_EXPR_CHECK(t)           TREE_CHECK (t, BIT_XOR_EXPR)
#define BIT_AND_EXPR_CHECK(t)           TREE_CHECK (t, BIT_AND_EXPR)
#define BIT_ANDTC_EXPR_CHECK(t)         TREE_CHECK (t, BIT_ANDTC_EXPR)
#define BIT_NOT_EXPR_CHECK(t)           TREE_CHECK (t, BIT_NOT_EXPR)
#define TRUTH_ANDIF_EXPR_CHECK(t)       TREE_CHECK (t, TRUTH_ANDIF_EXPR)
#define TRUTH_ORIF_EXPR_CHECK(t)        TREE_CHECK (t, TRUTH_ORIF_EXPR)
#define TRUTH_AND_EXPR_CHECK(t)         TREE_CHECK (t, TRUTH_AND_EXPR)
#define TRUTH_OR_EXPR_CHECK(t)          TREE_CHECK (t, TRUTH_OR_EXPR)
#define TRUTH_XOR_EXPR_CHECK(t)         TREE_CHECK (t, TRUTH_XOR_EXPR)
#define TRUTH_NOT_EXPR_CHECK(t)         TREE_CHECK (t, TRUTH_NOT_EXPR)
#define LT_EXPR_CHECK(t)                TREE_CHECK (t, LT_EXPR)
#define LE_EXPR_CHECK(t)                TREE_CHECK (t, LE_EXPR)
#define GT_EXPR_CHECK(t)                TREE_CHECK (t, GT_EXPR)
#define GE_EXPR_CHECK(t)                TREE_CHECK (t, GE_EXPR)
#define EQ_EXPR_CHECK(t)                TREE_CHECK (t, EQ_EXPR)
#define NE_EXPR_CHECK(t)                TREE_CHECK (t, NE_EXPR)
#define UNORDERED_EXPR_CHECK(t)         TREE_CHECK (t, UNORDERED_EXPR)
#define ORDERED_EXPR_CHECK(t)           TREE_CHECK (t, ORDERED_EXPR)
#define UNLT_EXPR_CHECK(t)              TREE_CHECK (t, UNLT_EXPR)
#define UNLE_EXPR_CHECK(t)              TREE_CHECK (t, UNLE_EXPR)
#define UNGT_EXPR_CHECK(t)              TREE_CHECK (t, UNGT_EXPR)
#define UNGE_EXPR_CHECK(t)              TREE_CHECK (t, UNGE_EXPR)
#define UNEQ_EXPR_CHECK(t)              TREE_CHECK (t, UNEQ_EXPR)
#define IN_EXPR_CHECK(t)                TREE_CHECK (t, IN_EXPR)
#define SET_LE_EXPR_CHECK(t)            TREE_CHECK (t, SET_LE_EXPR)
#define CARD_EXPR_CHECK(t)              TREE_CHECK (t, CARD_EXPR)
#define RANGE_EXPR_CHECK(t)             TREE_CHECK (t, RANGE_EXPR)
#define CONVERT_EXPR_CHECK(t)           TREE_CHECK (t, CONVERT_EXPR)
#define NOP_EXPR_CHECK(t)               TREE_CHECK (t, NOP_EXPR)
#define NON_LVALUE_EXPR_CHECK(t)        TREE_CHECK (t, NON_LVALUE_EXPR)
#define VIEW_CONVERT_EXPR_CHECK(t)      TREE_CHECK (t, VIEW_CONVERT_EXPR)
#define SAVE_EXPR_CHECK(t)              TREE_CHECK (t, SAVE_EXPR)
#define UNSAVE_EXPR_CHECK(t)            TREE_CHECK (t, UNSAVE_EXPR)
#define RTL_EXPR_CHECK(t)               TREE_CHECK (t, RTL_EXPR)
#define ADDR_EXPR_CHECK(t)              TREE_CHECK (t, ADDR_EXPR)
#define REFERENCE_EXPR_CHECK(t)         TREE_CHECK (t, REFERENCE_EXPR)
#define ENTRY_VALUE_EXPR_CHECK(t)       TREE_CHECK (t, ENTRY_VALUE_EXPR)
#define FDESC_EXPR_CHECK(t)             TREE_CHECK (t, FDESC_EXPR)
#define COMPLEX_EXPR_CHECK(t)           TREE_CHECK (t, COMPLEX_EXPR)
#define CONJ_EXPR_CHECK(t)              TREE_CHECK (t, CONJ_EXPR)
#define REALPART_EXPR_CHECK(t)          TREE_CHECK (t, REALPART_EXPR)
#define IMAGPART_EXPR_CHECK(t)          TREE_CHECK (t, IMAGPART_EXPR)
#define PREDECREMENT_EXPR_CHECK(t)      TREE_CHECK (t, PREDECREMENT_EXPR)
#define PREINCREMENT_EXPR_CHECK(t)      TREE_CHECK (t, PREINCREMENT_EXPR)
#define POSTDECREMENT_EXPR_CHECK(t)     TREE_CHECK (t, POSTDECREMENT_EXPR)
#define POSTINCREMENT_EXPR_CHECK(t)     TREE_CHECK (t, POSTINCREMENT_EXPR)
#define VA_ARG_EXPR_CHECK(t)            TREE_CHECK (t, VA_ARG_EXPR)
```

```
#define TRY_CATCH_EXPR_CHECK(t)          TREE_CHECK (t, TRY_CATCH_EXPR)
#define TRY_FINALLY_EXPR_CHECK(t)        TREE_CHECK (t, TRY_FINALLY_EXPR)
#define GOTO_SUBROUTINE_EXPR_CHECK(t)    TREE_CHECK (t, GOTO_SUBROUTINE_EXPR)
#define LABEL_EXPR_CHECK(t)              TREE_CHECK (t, LABEL_EXPR)
#define GOTO_EXPR_CHECK(t)               TREE_CHECK (t, GOTO_EXPR)
#define RETURN_EXPR_CHECK(t)             TREE_CHECK (t, RETURN_EXPR)
#define EXIT_EXPR_CHECK(t)               TREE_CHECK (t, EXIT_EXPR)
#define LOOP_EXPR_CHECK(t)               TREE_CHECK (t, LOOP_EXPR)
#define LABELED_BLOCK_EXPR_CHECK(t)      TREE_CHECK (t, LABELED_BLOCK_EXPR)
#define EXIT_BLOCK_EXPR_CHECK(t)         TREE_CHECK (t, EXIT_BLOCK_EXPR)
#define EXPR_WITH_FILE_LOCATION_CHECK(t)         TREE_CHECK (t, EXPR_WITH_FILE_LOCATION)
#define SWITCH_EXPR_CHECK(t)             TREE_CHECK (t, SWITCH_EXPR)
#define EXC_PTR_EXPR_CHECK(t)            TREE_CHECK (t, EXC_PTR_EXPR)
#define SRCLOC_CHECK(t)                  TREE_CHECK (t, SRCLOC)
#define SIZEOF_EXPR_CHECK(t)             TREE_CHECK (t, SIZEOF_EXPR)
#define ARROW_EXPR_CHECK(t)              TREE_CHECK (t, ARROW_EXPR)
#define ALIGNOF_EXPR_CHECK(t)            TREE_CHECK (t, ALIGNOF_EXPR)
#define EXPR_STMT_CHECK(t)               TREE_CHECK (t, EXPR_STMT)
#define COMPOUND_STMT_CHECK(t)           TREE_CHECK (t, COMPOUND_STMT)
#define DECL_STMT_CHECK(t)               TREE_CHECK (t, DECL_STMT)
#define IF_STMT_CHECK(t)                 TREE_CHECK (t, IF_STMT)
#define FOR_STMT_CHECK(t)                TREE_CHECK (t, FOR_STMT)
#define WHILE_STMT_CHECK(t)              TREE_CHECK (t, WHILE_STMT)
#define DO_STMT_CHECK(t)                 TREE_CHECK (t, DO_STMT)
#define RETURN_STMT_CHECK(t)             TREE_CHECK (t, RETURN_STMT)
#define BREAK_STMT_CHECK(t)              TREE_CHECK (t, BREAK_STMT)
#define CONTINUE_STMT_CHECK(t)           TREE_CHECK (t, CONTINUE_STMT)
#define SWITCH_STMT_CHECK(t)             TREE_CHECK (t, SWITCH_STMT)
#define GOTO_STMT_CHECK(t)               TREE_CHECK (t, GOTO_STMT)
#define LABEL_STMT_CHECK(t)              TREE_CHECK (t, LABEL_STMT)
#define ASM_STMT_CHECK(t)                TREE_CHECK (t, ASM_STMT)
#define SCOPE_STMT_CHECK(t)              TREE_CHECK (t, SCOPE_STMT)
#define FILE_STMT_CHECK(t)               TREE_CHECK (t, FILE_STMT)
#define CASE_LABEL_CHECK(t)              TREE_CHECK (t, CASE_LABEL)
#define STMT_EXPR_CHECK(t)               TREE_CHECK (t, STMT_EXPR)
#define COMPOUND_LITERAL_EXPR_CHECK(t)   TREE_CHECK (t, COMPOUND_LITERAL_EXPR)
#define CLEANUP_STMT_CHECK(t)            TREE_CHECK (t, CLEANUP_STMT)
```

위 정의들은 gencheck 바이너리에 의해서 자동으로 생성되는 부분이다.

```
#define TYPE_CHECK(tree)    TREE_CLASS_CHECK  (tree, 't')
#define DECL_CHECK(tree)    TREE_CLASS_CHECK  (tree, 'd')
#define CST_CHECK(tree)     TREE_CLASS_CHECK  (tree, 'c')
```

현재 tree 가 TYPE 혹은 DECL, CST 인지를 확인하는 정의들이다.

```
#define TREE_TYPE(NODE) ((NODE)->common.type)
```

expression 인 모든 node 내에서 이것은 expression 의 date type 을 나타냅니다.
POINTER_TYPE node 에서는 이것은 포인터를 가르키는 type 입니다.
ARRAY_TYPE node 에서는 이것은 element 들의 type 입니다.
VECTOR_TYPE node 에서는 이것은 element 들의 type 입니다.

```
#define TYPE_HASH(TYPE) ((size_t) (TYPE) & 0777777)
```

이것은 어떻게 primitive 혹은 already-canonicalized types' hash code 들이 만들어 졌는지를 말합니다.

```
#define TREE_CHAIN(NODE) ((NODE)->common.chain)
```

Node 들은 많은 목적을 위해 함께 연결됩니다.
Type 들은 debugger 에 output 하기 위해 그들을 기록함으로써 함께 연결된다. (함수 'chain_type' 를 보세요).
같은 scope 내의 decl 들은 scope 의 content 들을 기록하기 위해 함께 chain 됩니다.
성공적인 statement 들을 위한 statement node 들도 함께 연결되기 위해 사용된다. 종종 그러한 것의 list 들은 함께 연결된 TREE_LIST node 들로 나타나기도 한다.

```
#define STRIP_NOPS(EXP)                         \
  while ((TREE_CODE (EXP) == NOP_EXPR                \
      || TREE_CODE (EXP) == CONVERT_EXPR             \
      || TREE_CODE (EXP) == NON_LVALUE_EXPR)         \
     && TREE_OPERAND (EXP, 0) != error_mark_node         \
     && (TYPE_MODE (TREE_TYPE (EXP))             \
         == TYPE_MODE (TREE_TYPE (TREE_OPERAND (EXP, 0))))) \
    (EXP) = TREE_OPERAND (EXP, 0)
```

Tree 로써 주어진 표현식에서 machine mode 를 변경하지 않는 어떠한 NON_LVALUE_EXPR 들과 NOP_EXPR 들을 짜른다.

```
#define STRIP_SIGN_NOPS(EXP) \
  while ((TREE_CODE (EXP) == NOP_EXPR                \
      || TREE_CODE (EXP) == CONVERT_EXPR             \
      || TREE_CODE (EXP) == NON_LVALUE_EXPR)         \
     && TREE_OPERAND (EXP, 0) != error_mark_node         \
     && (TYPE_MODE (TREE_TYPE (EXP))             \
         == TYPE_MODE (TREE_TYPE (TREE_OPERAND (EXP, 0))))  \
     && (TREE_UNSIGNED (TREE_TYPE (EXP))             \
         == TREE_UNSIGNED (TREE_TYPE (TREE_OPERAND (EXP, 0))))) \
    (EXP) = TREE_OPERAND (EXP, 0)
```

STRIP_NOPS 와 비슷하지만 signedness chage 에 대해 또한 수행하지 않는다.

```
#define STRIP_TYPE_NOPS(EXP) \
  while ((TREE_CODE (EXP) == NOP_EXPR                \
      || TREE_CODE (EXP) == CONVERT_EXPR             \
      || TREE_CODE (EXP) == NON_LVALUE_EXPR)         \
     && TREE_OPERAND (EXP, 0) != error_mark_node         \
     && (TREE_TYPE (EXP)                      \
         == TREE_TYPE (TREE_OPERAND (EXP, 0))))     \
    (EXP) = TREE_OPERAND (EXP, 0)
```

STRIP_NOPS 와 비슷하지만 TREE_TYPE 를 또한 변경하지 않는다.

```
#define INTEGRAL_TYPE_P(TYPE)  \
  (TREE_CODE (TYPE) == INTEGER_TYPE || TREE_CODE (TYPE) == ENUMERAL_TYPE  \
   || TREE_CODE (TYPE) == BOOLEAN_TYPE || TREE_CODE (TYPE) == CHAR_TYPE)
```

0 이 아닐 경우 TYPE 은 integral type 을 나타냅니다. 우리는 여기서 COMPLEX type 들을 포함하지 않는다는 것을 주목하십시오.

```
#define FLOAT_TYPE_P(TYPE)      \
  (TREE_CODE (TYPE) == REAL_TYPE    \
   || (TREE_CODE (TYPE) == COMPLEX_TYPE \
       && TREE_CODE (TREE_TYPE (TYPE)) == REAL_TYPE))
```

0 이 아닐 경우 TYPE 이 complex floating-point type 들을 포함하는 floating-point type 을 나타냅니다.

```
#define AGGREGATE_TYPE_P(TYPE) \
  (TREE_CODE (TYPE) == ARRAY_TYPE || TREE_CODE (TYPE) == RECORD_TYPE \
   || TREE_CODE (TYPE) == UNION_TYPE || TREE_CODE (TYPE) == QUAL_UNION_TYPE \
   || TREE_CODE (TYPE) == SET_TYPE)
```

0 이 아닐 경우 TYPE 은 integral type 을 나타냅니다. 우리는 여기서 COMPLEX type 들을 포함하지 않는다는 것을 주목하십시오.

```
#define POINTER_TYPE_P(TYPE) \
  (TREE_CODE (TYPE) == POINTER_TYPE || TREE_CODE (TYPE) == REFERENCE_TYPE)
```

만약 TYPE 이 unbounded pointer 혹은 unbounded reference type 을 표현하면 0 이 아닌 값을. (반드시 INDIRECT_TYPE_P 로 재 이름 설정되어야 한다.)

```
#define BOUNDED_INDIRECT_TYPE_P(TYPE) \
  (TREE_CODE (TYPE) == RECORD_TYPE && TREE_TYPE (TYPE))
```

만약 TYPE 이 bounded pointer 혹은 bounded reference type 을 나타낸다면 0 이 아닌 값.

```
#define BOUNDED_POINTER_TYPE_P(TYPE) \
  (BOUNDED_INDIRECT_TYPE_P (TYPE) \
   && TREE_CODE (TYPE_BOUNDED_SUBTYPE (TYPE)) == POINTER_TYPE)
```

0 이 아닐 경우 TYPE 은 bounded pointer type 을 나타냅니다.

```
#define BOUNDED_REFERENCE_TYPE_P(TYPE) \
  (BOUNDED_INDIRECT_TYPE_P (TYPE) \
   && TREE_CODE (TYPE_BOUNDED_SUBTYPE (TYPE)) == REFERENCE_TYPE)
```

만약 TYPE 이 bounded reference type 을 나타낸다면 0 이 아닌 값. Bounded reference type 들은 두가지 종류의 사용법을 가지고 있다: (1) Reference 가 그것의 마지막 field 로써 불확정 길이의 배열을 가지고 있는 variable-length RECORD_TYPE 에 위치에 있을 때. 다른 object 들에 대해, reference 들의 주소가 변경될 수 없기 때문에 reference 가 위치한 시기에 bound 들을 검사하는 것이 충분하며 나중에 reference 의 사용이 모두 안전 하다고 가정한다. (2) Reference supertype 이 subtype object 에 위치해 있을 때. Bound 들은 완료된 object 의 실제 크기를 "기억하므로", reference 주소의 차후의 upcast 들은 적당히 검사될 것이다. (이것이 C++ 에서도 유효 하나요?).

```
#define MAYBE_BOUNDED_INDIRECT_TYPE_P(TYPE) \
  (POINTER_TYPE_P (TYPE) || BOUNDED_INDIRECT_TYPE_P (TYPE))
```

만약 TYPE 이 pointer 혹은 reference type 을 나타낸다면 0 이 아닌 값을 가진다. bounded 혹은 unbounded 이든 상관없다.

```
#define MAYBE_BOUNDED_POINTER_TYPE_P(TYPE) \
  (TREE_CODE (TYPE) == POINTER_TYPE || BOUNDED_POINTER_TYPE_P (TYPE))
```

만약 TYPE 이 pointer type 을 나타낸다면 값이 0 이 아닌 값. bounded 혹은 unbounded 이든 상관없다.

```
#define MAYBE_BOUNDED_REFERENCE_TYPE_P(TYPE) \
  (TREE_CODE (TYPE) == REFERENCE_TYPE || BOUNDED_REFERENCE_TYPE_P (TYPE))
```

만약 TYPE 이 reference type 을 나타낸다면 값이 0 이 아닌 값. bounded 혹은 unbounded 이든 상관없다.

```
#define COMPLETE_TYPE_P(NODE) (TYPE_SIZE (NODE) != NULL_TREE)
```

만약 이 type 이 완료된 type 이면 0 이 아닌 값.

```
#define VOID_TYPE_P(NODE) (TREE_CODE (NODE) == VOID_TYPE)
```

만약 이 type 이 (가능하다 판단되는) void type 이면 0 이 아닌 값.

```
#define COMPLETE_OR_VOID_TYPE_P(NODE) \
  (COMPLETE_TYPE_P (NODE) || VOID_TYPE_P (NODE))
```

만약 이 type 이 완료했거나 cv void 이면 0 이 아닌 값.

```
#define COMPLETE_OR_UNBOUND_ARRAY_TYPE_P(NODE) \
  (COMPLETE_TYPE_P (TREE_CODE (NODE) == ARRAY_TYPE ? TREE_TYPE (NODE) : (NODE)))
```

만약 이 type 이 완료했거나 경계가 지정되지 않은 배열이면 0 이 아닌 값.

```
#define TYPE_P(TYPE)    (TREE_CODE_CLASS (TREE_CODE (TYPE)) == 't')
```

0 이 아닐 경우 TYPE 은 type 을 나타냅니다.

```
#define TREE_ADDRESSABLE(NODE) ((NODE)->common.addressable_flag)
```

VAR_DECL node 에서는, 0 이 아닌 값은 이것의 주소가 필요함을 의미한다. 그래서 이것은 register 내에 있을 수 없다.
FUNCTION_DECL 에서는, 0 이 아닌 값은 그것의 주소가 필요함을 의미한다. 그래서 그것이 inline function 이라 할지라도 반드시 컴파일되어야 한다.
FIELD_DECL node 에서는, 이 field 의 주소를 만드는 것을 프로그래머가 허락했음을 의미한다. 이것은 목적들 (purposes) 를 alias 하는데 사용된다: record_component_aliases 를 보라.
CONSTRUCTOR node 에서는, 만들어진 object 가 반드시 메모리내에 있어야함을 의미한다.
LABEL_DECL node 에서는, 이 label 을 위한 goto 가 stack level 들을 복구하기 위한 모든 binding contour 들의 바깥쪽 장소에서 보여져야 함을 의미한다.
...._TYPE node 에서는, 이 type 의 object 들은 반드시 전체적으로 주소화가 가능해야 (addressable) 함을 의미한다. 예를 들면, 이것은 이 object 의 조각들이 register parameter 내로 갈수 없음을 의미한다.
IDENTIFIER_NODE 들에서는, 이것의 이름을 위한 몇몇 외부 decl 가 그것의 주소를 가지고 있음을 의미한다. 이것은 inline 함수들에 대한 문제이다.

```
#define TREE_STATIC(NODE) ((NODE)->common.static_flag)
```

VAR_DECL 에서 0 이 아닌 값은 'static storage 를 할당한다' 를 의미합니다.
FUNCTION_DECL 에서 함수가 정의되었을 경우 0 이 아닌 값을 가짐.
CONSTRUCTOR 에서 0 이 아닌 값은 'static storage 를 할당한다' 를 의미합니다.

```
#define CLEANUP_EH_ONLY(NODE) ((NODE)->common.static_flag)
```

TARGET_EXPR 혹은 WITH_CLEANUP_EXPR, CLEANUP_STMT, block 의 cleanup list 의 element 에서는, 그것의 scope 의 보통 exit 상에서는 아닌, exception 이 던져진 경우에만 타당한 cleanup 이 실행되어야 함을 의미한다.

```
#define TREE_NO_UNUSED_WARNING(NODE) ((NODE)->common.static_flag)
```

CONVERT_EXPR 혹은 NOP_EXPR, COMPOUND_EXPR 에서 이것은 해당 node 가 묵시적으로 생성되었고 "unused value" 경고문을 발생시키지 말아야함을 의미한다.

```
#define TREE_VIA_VIRTUAL(NODE) ((NODE)->common.static_flag)
```

TREE_LIST 혹은 TREE_VEC node 에 대한 0 이 아닌 값은 유도 chain 이 'virtual' 선언을 통한 것임을 의미한다.

```
#define TREE_CONSTANT_OVERFLOW(NODE) ((NODE)->common.static_flag)
```

INTEGER_CST 혹은 REAL_CST, COMPLEX_CST, VECTOR_CST 에서 이것은 folding 과정에서 overflow 가 있었음을 의미한다. 이것은 TREE_OVERFLOW 와 뚜렷하게 구분이 되는데, ANSI C 는 상수 표현식에서 overflow 가 발생하였을 때, 진단 (diagnostic) 을 요구하기 때문이다.

```
#define TREE_SYMBOL_REFERENCED(NODE) \
  (IDENTIFIER_NODE_CHECK (NODE)->common.static_flag)
```

IDENTIFIER_NODE 에서 이것은 assemble_name 가 인자로써 이 문자열로 호출 되었음을 의미한다.

```
#define TREE_OVERFLOW(NODE) ((NODE)->common.public_flag)
```

INTEGER_CST 혹은 REAL_CST, COMPLEX_CST, VECTOR_CST 에서 이것은 folding 과정에서 overflow 가 있었고 이 하위표현식에 대해 어떠한 경고도 발생되지 않았음을 의미한다. TREE_OVERFLOW 는 TREE_CONSTANT_OVERFLOW 를 내포하지만, 반대로는 아니다.

```
#define TREE_PUBLIC(NODE) ((NODE)->common.public_flag)
```

VAR_DECL 혹은 FUNCTION_DECL 에서 0 이 아닌 경우, 이 module 외부에서 이름에 접근할 수 있음을 의미합니다. IDENTIFIER_NODE 에서, 0 이 아닌 것은 이 모듈 외부로부터 접근 가능한 외부 선언이 내부 scope 에 이 이름과 같은 것이 이전에 보였음을 의미한다.

```
#define TREE_VIA_PUBLIC(NODE) ((NODE)->common.public_flag)
```

TREE_LIST 혹은 TREE_VEC node 에 대해 0 이 아닌 값은 base class 로의 경로가 public 으로써 base class 로부터 public field 들을 보전하고 있는 'public' 선언을 통해있음을 의미한다.

```
#define TREE_VIA_PRIVATE(NODE) ((NODE)->common.private_flag)
```

전과 같으며, 'private' 선언에 대한 것.

```
#define TREE_VIA_PROTECTED(NODE) ((NODE)->common.protected_flag)
```

TREE_LIST 혹은 TREE_VEC node 에 대해 0 이 아닌 값은 base class 로의 경로가 protected 으로써 base class 로부터 protected field 들을 보전하고 있는 'protected' 선언을 통해있음을 의미한다. OVERLOADED.

```
#define TREE_SIDE_EFFECTS(NODE) ((NODE)->common.side_effects_flag)
```

어떤 표현식에서 0 이 아닌 값은 그것이 부과적인 영향을 가지고 있거나 전체 표현식이 다른 값을 생산할수 있도록 하는 재평가를 가지고 있음을 의미한다. 이것은 어떠한 하위표현식이 함수 호출 혹은 부과적인 영향, volatile 변수로의 reference 라면 설정된다.
...._DECL 에서 이것은 만약 선언이 'volatile' 을 말하는 경우만 설정된다.

```
#define TREE_THIS_VOLATILE(NODE) ((NODE)->common.volatile_flag)
```

값이 0 이 아님은 이 표현식이 C 에서는 volatile 임을 의미한다: 그것의 주소는 반드시 type 'volatile WHATEVER *' 형태여야 한다. 다르게 말해서, 선언된 item 은 volatile 로 인정받았다는 것이다. 이것은 _DECL node 들과 _REF node 들에서 사용된다.

...._TYPE node 에서 이 type 은 volatile 로 인정받았다. 하지만 해당 node 가 type 이라면 이 macro 대신 TYPE_VOLATILE 를 사용하라 왜냐하면 결과적으로 우리는 다른 bit 를 만들 것이기 때문이다.

만약 이 bit 가 표현식에서 설정되었다면, 그것은 TREE_SIDE_EFFECTS 이다.

```
#define TREE_READONLY(NODE) ((NODE)->common.readonly_flag)
```

VAR_DECL 혹은 PARM_DECL, FIELD_DECL, ..._REF node 의 어떤 종류에서 값이 0 이 아님은 이것은 assignment 의 lhs 가 아닐것임을 의미한다. ...._TYPE node 에서는 이 type 이 const 로 인정받았음을 의미 (하지만 해당 node 가 type 일 때 이 macro 대신 macro TYPE_READONLY 는 반드시 사용되어야 한다.).

```
#define TREE_READONLY_DECL_P(NODE) (TREE_READONLY (NODE) && DECL_P (NODE))
```

만약 NODE 가 TREE_READONLY 설정을 가진 _DECL 이라면 0 이 아닌 값.

```
#define TREE_CONSTANT(NODE) ((NODE)->common.constant_flag)
```

표현식의 값이 상수이다.
항상 모든 ...._CST node 들에서 보인다. 또한 값이 상수일 경우, arithmetic 표현식 혹은 ADDR_EXPR, CONSTRUCTOR 에서도 보일 수 있다.

```
#define TREE_UNSIGNED(NODE) ((NODE)->common.unsigned_flag)
```

INTEGER_TYPE 혹은 ENUMERAL_TYPE node 들에서는 unsigned type 을 의미한다.
FIELD_DECL node 들에서는 unsigned bit field 를 의미한다. 같은 bit 가 DECL_BUILT_IN_NONANSI 처럼 함수들내에서 사용된다.

```
#define TYPE_TRAP_SIGNED(NODE) \
  (flag_trapv && ! TREE_UNSIGNED (TYPE_CHECK (NODE)))
```

설명 없음.

```
#define TREE_ASM_WRITTEN(NODE) ((NODE)->common.asm_written_flag)
```

VAR_DECL 에서의 값이 0 이 아님은 어셈블러 code 가 이미 쓰였음 (written) 을 의미한다.
FUNCTION_DECL 에서의 값이 0 이 아님은 함수가 이미 컴파일되었음을 의미한다. 이것은 inline 함수내에서 유용할 수 있는데, 별개적으로 컴파일될 필요가 없기 때문이다.
RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE, ENUMERAL_TYPE 에서의 값이 0 이 아님은 해당 type 을 위한 sdb debugging info 가 이미 쓰였음을 의미할 경우를 말한다.
BLOCK node 에서는, 만약 reorder_blocks 이 이미 이 block 에서 보여졌다면 값이 0 이 아닌 것을 가진다.

```
#define TREE_USED(NODE) ((NODE)->common.used_flag)
```

만약 이름이 그것의 scope 내에서 사용되었다면 _DECL 에서는 값이 0 아님. expr node 에서 값이 0 이 아님은 값이 사용되지 않았을 경우 경고문을 금지 함을 의미한다.
IDENTIFIER_NODE 들에서 이 이름을 위한 몇몇 외부 decl 이 사용되었음을 의미한다.

```
#define TREE_NOTHROW(NODE) ((NODE)->common.nothrow_flag)
```

FUNCTION_DECL 에서, 값이 0 이 아님은 함수로의 호출이 exception 을 던질 수 없 음을 의미한다.
CALL_EXPR 에서 0 이 아님은 호출을 던질 수 없음을 의미한다.

```
#define TYPE_ALIGN_OK(NODE) (TYPE_CHECK (NODE)->common.nothrow_flag)
```

Type 에서 값이 0 이 아님은 type 의 모든 object 들이 language 혹은 front-end 에 의해 적당히 align 될 것을 보장받으며, 그래서 우리가 이 type 의 MEM 이 그것이 어떤것인지 나타나지 않아도 적어도 해당 type 의 alignment 로 align 되었음을 지시할 수 있다. 예를 들면, tag field 가 이것이 더 generic type 의 more-aligned variant 의 object 임을 보여줘야 하는 object-oriented 언어들에서 우리는 이것을 볼 수 있다.

```
#define TREE_PRIVATE(NODE) ((NODE)->common.private_flag)
```

C++ 내의 class 에서 사용됩니다.

```
#define TREE_PROTECTED(NODE) ((NODE)->common.protected_flag)
```

C++ 내의 class 에서 사용됩니다.
BLOCK node 에서, 이것은 BLOCK_HANDLER_BLOCK 이다.

```
#define TREE_BOUNDED(NODE) ((NODE)->common.bounded_flag)
```

..._TYPE node 에서 값이 0 이 아님은 type 의 크기와 layout, (혹은 그것의 argument 들의 크기와 layous 과/혹은 FUNCTION_TYPE 혹은 METHOD_TYPE 의 경우에서의 return 값) 은 pointer bound 들의 존재로 인해 변경되었음을 의미한다. node 가 type 일 경우 이 macro 대신 TYPE_BOUNDED 를 사용하라. 왜냐하면 결과적으로 우리는 다른 bit 들을 만들 것이기 때문이다. TYPE_BOUNDED 는 이 type 이 bounded indirect type 인 것을 의미하지 않는다. – 그것을 검사하기 위해서는 BOUNDED_POINTER_TYPE_P, BOUNDED_REFERENCE_TYPE_P, BOUNDED_INDIRECT_TYPE_P 를 사용하라.

FUNCTION_DECL 에서의 값이 0 이 아님은 그것의 argument 의 어떤것의 크기와 layout 과/혹은 return 값이 pointer bound 들의 존재로 인해 변경되었음을 의미한다. 이 값은 함수가 묵시적으로 선언되어 후에 pointer arg 들과 함께 호출되거나, variable argument list 와 함께 선언되어 variable argument list 내의 pointer 값과 함께 후에 호출될 수 있기에 TYPE_BOUNDED (TREE_TYPE (fundecl)) 의 값과 다를 수 있다.

VAR_DECL 혹은 PARM_DECL, FIELD_DECL, TREE_BOUNDED 는 decl 의 type 의 BOUNDED_POINTER_TYPE_P 의 값과 매칭된다.

CONSTRUCTOR 혹은 다른 표현식에서 값이 0 이 아님은 값이 bounded pointer 임을 의미한다. 이것은 BOUNDED_POINTER_TYPE_P (TREE_TYPE (EXP)) 를 가진 표현식 EXP 의 boundedness 를 결정하는데 불충분한데, 왜냐하면 우리는 pointer 의 bound 들을 보존하는 방식에서 alignment boundary 까지의 rounding up 을 위한 정수를 임시적으로 cast 될 수 있음을 pointer 에 허락하기 때문이다.

IDENTIFIER_NODE 에서 값이 0 이 아님은 이름이 BP_PREFIX (varasm.c 를 참조) 로 접두사화되었음을 의미한다. 이것은 해당 return type 과/혹은 argument type(들) 을 위한 pointer(들) 가 bound 한 함수의 DECL_ASSEMBLER_NAME 일 경우에 발생한다.

```
#define TREE_DEPRECATED(NODE) ((NODE)->common.deprecated_flag)
```

만약 INDENTIFIER_NODE 내에서 이름의 사용이 __attribute__((deprecated)) 에 의해 deprecated feature 로써 정의되어 진다면 0 이 아닌 값을 가집니다.

```
#define TREE_LANG_FLAG_0(NODE) ((NODE)->common.lang_flag_0)
#define TREE_LANG_FLAG_1(NODE) ((NODE)->common.lang_flag_1)
#define TREE_LANG_FLAG_2(NODE) ((NODE)->common.lang_flag_2)
#define TREE_LANG_FLAG_3(NODE) ((NODE)->common.lang_flag_3)
#define TREE_LANG_FLAG_4(NODE) ((NODE)->common.lang_flag_4)
#define TREE_LANG_FLAG_5(NODE) ((NODE)->common.lang_flag_5)
#define TREE_LANG_FLAG_6(NODE) ((NODE)->common.lang_flag_6)
```

아래의 flag 들은 각 언어의 front end 가 내부적으로 사용할 용도로 이용 할 수 있습니다.

3.2.2 tree_int_cst 구조체

상수들을 표현하는 node 들을 위한 추가적인 field 와 accessor 들을 정의한다.

```
#define TREE_INT_CST(NODE) (INTEGER_CST_CHECK (NODE)->int_cst.int_cst)
#define TREE_INT_CST_LOW(NODE) (TREE_INT_CST (NODE).low)
#define TREE_INT_CST_HIGH(NODE) (TREE_INT_CST (NODE).high)
```

INTEGER_CST node 에서. 다음의 두개가 하나의 2-word 정수를 만든다. 만약 data type 이 sign 으로 되어 있으면, 값은 그것들이 전부 사용되지 않고 있다 하더라도 sign-extended 된 2 word 들이다. 2 word 보다 작은 unsign 상수라면, extra bit 들은 0 이다.

```
#define INT_CST_LT(A, B)                \
  (TREE_INT_CST_HIGH (A) < TREE_INT_CST_HIGH (B)    \
   || (TREE_INT_CST_HIGH (A) == TREE_INT_CST_HIGH (B)   \
       && TREE_INT_CST_LOW (A) < TREE_INT_CST_LOW (B)))
```

설명 없음.

```
#define INT_CST_LT_UNSIGNED(A, B)               \
  (((unsigned HOST_WIDE_INT) TREE_INT_CST_HIGH (A)      \
    < (unsigned HOST_WIDE_INT) TREE_INT_CST_HIGH (B))       \
   || (((unsigned HOST_WIDE_INT) TREE_INT_CST_HIGH (A)      \
        == (unsigned HOST_WIDE_INT) TREE_INT_CST_HIGH (B))  \
       && TREE_INT_CST_LOW (A) < TREE_INT_CST_LOW (B)))
```

설명 없음.

3.2.3 tree_real_cst 구조체

```
#define TREE_CST_RTL(NODE) (CST_OR_CONSTRUCTOR_CHECK (NODE)->real_cst.rtl)
```

REAL_CST 와 STRING_CST, COMPLEX_CST, VECTOR_CST node 들, CONSTRUCTOR node 들 과 (이제 막 선언된 것보다) 일반적으로 label 들이 주어진 모든 종류의 상수들.

```
#define TREE_REAL_CST(NODE) (REAL_CST_CHECK (NODE)->real_cst.real_cst)
```

REAL_CST node 에서 우리는 'double' 혹은 long 들의 배열로써 실수값을 나타낼 수 있다.

3.2.4 tree_string 구조체

```
#define TREE_STRING_LENGTH(NODE) (STRING_CST_CHECK (NODE)->string.length)
```

STRING_CST 에서 문자열의 길이를 나타낸다.

```
#define TREE_STRING_POINTER(NODE) (STRING_CST_CHECK (NODE)->string.pointer)
```

STRING_CST 에서 문자열 pointer 를 가르킨다.

3.2.5 tree_complex 구조체

```
#define TREE_REALPART(NODE) (COMPLEX_CST_CHECK (NODE)->complex.real)
```

COMPLEX_CST node 에서 실수 부분을 나타낸다.

```
#define TREE_IMAGPART(NODE) (COMPLEX_CST_CHECK (NODE)->complex.imag)
```

COMPLEX_CST node 에서 허수 부분을 나타낸다.

3.2.6 tree_vector 구조체

VECTOR_CST node 에서.

```
#define TREE_VECTOR_CST_ELTS(NODE) (VECTOR_CST_CHECK (NODE)->vector.elements)
```

3.2.7 tree_identifier 구조체

```
#define IDENTIFIER_LENGTH(NODE) \
  (IDENTIFIER_NODE_CHECK (NODE)->identifier.id.len)
#define IDENTIFIER_POINTER(NODE) \
  ((const char *) IDENTIFIER_NODE_CHECK (NODE)->identifier.id.str)
```

몇몇 특정-목적의 tree node 들을 위한 field 와 연결자 매크로를 정의합니다.

```
#define HT_IDENT_TO_GCC_IDENT(NODE) \
  ((tree) ((char *) (NODE) - sizeof (struct tree_common)))
#define GCC_IDENT_TO_HT_IDENT(NODE) (&((struct tree_identifier *) (NODE))->id)
```

Hash table 식별자 포인터를 tree_identifier 포인터로 번역합니다. 반대로도 정의합니다.

3.2.8 tree_list 구조체

TREE_LIST node 내에서 사용되는 연결자 매크로.

```
#define TREE_PURPOSE(NODE) (TREE_LIST_CHECK (NODE)->list.purpose)
#define TREE_VALUE(NODE) (TREE_LIST_CHECK (NODE)->list.value)
```

3.2.9 tree_vec 구조체

TREE_VEC node 내에서 사용되는 연결자 매크로.

```
#define TREE_VEC_LENGTH(NODE) (TREE_VEC_CHECK (NODE)->vec.length)
#define TREE_VEC_ELT(NODE,I) (TREE_VEC_CHECK (NODE)->vec.a[I])
#define TREE_VEC_END(NODE) \
  ((void) TREE_VEC_CHECK (NODE), &((NODE)->vec.a[(NODE)->vec.length]))
```

3.2.10 tree_exp 구조체

expression 들을 나타내는 몇몇 node 들을 위한 field 와 연결자 매크로를 정의합니다.

```
#define SAVE_EXPR_CONTEXT(NODE) TREE_OPERAND (SAVE_EXPR_CHECK (NODE), 1)
#define SAVE_EXPR_RTL(NODE) (*(rtx *) &SAVE_EXPR_CHECK (NODE)->exp.operands[2])
#define SAVE_EXPR_NOPLACEHOLDER(NODE) TREE_UNSIGNED (SAVE_EXPR_CHECK (NODE))
```

SAVE_EXPR node 용.

```
#define SAVE_EXPR_PERSISTENT_P(NODE) TREE_ASM_WRITTEN (SAVE_EXPR_CHECK (NODE))
```

만약 SAVE_EXPR 의 값이 일반적인 code 내에서도 handler 내에서도 발생하더라도 유지되어야 한다면 0 이 아닌 값을 갖는다. (보통, handler 내에서, 모든 SAVE_EXPR 들이 저장되지 않으며, 즉 그러한 값들은 재계산되어야 함을 의미한다.)

```
#define RTL_EXPR_SEQUENCE(NODE) \
  (*(rtx *) &RTL_EXPR_CHECK (NODE)->exp.operands[0])
#define RTL_EXPR_RTL(NODE) (*(rtx *) &RTL_EXPR_CHECK (NODE)->exp.operands[1])
```

RTL_EXPR node 용.

```
#define WITH_CLEANUP_EXPR_RTL(NODE) \
  (*(rtx *) &WITH_CLEANUP_EXPR_CHECK (NODE)->exp.operands[2])
```

WITH_CLEANUP_EXPR node 용.

```
#define CONSTRUCTOR_ELTS(NODE) TREE_OPERAND (CONSTRUCTOR_CHECK (NODE), 1)
```

CONSTRUCTOR node 용.

```
#define TREE_OPERAND(NODE, I) (EXPR_CHECK (NODE)->exp.operands[I])
#define TREE_COMPLEXITY(NODE) (EXPR_CHECK (NODE)->exp.complexity)
```

보통의 표현식 node 들에서.

```
#define LABELED_BLOCK_LABEL(NODE) \
  TREE_OPERAND (LABELED_BLOCK_EXPR_CHECK (NODE), 0)
#define LABELED_BLOCK_BODY(NODE) \
  TREE_OPERAND (LABELED_BLOCK_EXPR_CHECK (NODE), 1)
```

LABELED_BLOCK_EXPR node 용.

```
#define EXIT_BLOCK_LABELED_BLOCK(NODE) \
  TREE_OPERAND (EXIT_BLOCK_EXPR_CHECK (NODE), 0)
#define EXIT_BLOCK_RETURN(NODE) TREE_OPERAND (EXIT_BLOCK_EXPR_CHECK (NODE), 1)
```

EXIT_BLOCK_EXPR node 용.

```
#define LOOP_EXPR_BODY(NODE) TREE_OPERAND (LOOP_EXPR_CHECK (NODE), 0)
```

LOOP_EXPR node 에서.

```
#define EXPR_WFL_EMIT_LINE_NOTE(NODE) \
  (EXPR_WITH_FILE_LOCATION_CHECK (NODE)->common.public_flag)
#define EXPR_WFL_NODE(NODE) \
  TREE_OPERAND (EXPR_WITH_FILE_LOCATION_CHECK (NODE), 0)
#define EXPR_WFL_FILENAME_NODE(NODE) \
  TREE_OPERAND (EXPR_WITH_FILE_LOCATION_CHECK (NODE), 1)
#define EXPR_WFL_FILENAME(NODE) \
  IDENTIFIER_POINTER (EXPR_WFL_FILENAME_NODE (NODE))
```

EXPR_WITH_FILE_LOCATION node 용.

```
#define EXPR_WFL_LINECOL(NODE) (EXPR_CHECK (NODE)->exp.complexity)
#define EXPR_WFL_LINENO(NODE) (EXPR_WFL_LINECOL (NODE) >> 12)
#define EXPR_WFL_COLNO(NODE) (EXPR_WFL_LINECOL (NODE) & 0xfff)
#define EXPR_WFL_SET_LINECOL(NODE, LINE, COL) \
  (EXPR_WFL_LINECOL(NODE) = ((LINE) << 12) | ((COL) & 0xfff))
```

??? Java 는 모든 expression 에서 이거을 사용합니다.

3.2.11 tree_block 구조체

```
#define BLOCK_VARS(NODE) (BLOCK_CHECK (NODE)->block.vars)
#define BLOCK_SUBBLOCKS(NODE) (BLOCK_CHECK (NODE)->block.subblocks)
#define BLOCK_SUPERCONTEXT(NODE) (BLOCK_CHECK (NODE)->block.supercontext)
```

BLOCK node 에서 사용되는 것.

```
#define BLOCK_CHAIN(NODE) TREE_CHAIN (BLOCK_CHECK (NODE))
#define BLOCK_ABSTRACT_ORIGIN(NODE) (BLOCK_CHECK (NODE)->block.abstract_origin)
#define BLOCK_ABSTRACT(NODE) (BLOCK_CHECK (NODE)->block.abstract_flag)
```

알림: 이것이 변할 때, chainon 혹은 nreverse 를 사용하는 장소를 찾을 수 있도록 확실히 하십시오.

```
#define BLOCK_HANDLER_BLOCK(NODE) \
  (BLOCK_CHECK (NODE)->block.handler_block_flag)
```

값이 0 이 아님은 이 block 이 BLOCK_VARS slot 에 나열되어 있는 exception 들을 다루기 위해 준비되었음을 의미한다.

```
#define BLOCK_NUMBER(NODE) (BLOCK_CHECK (NODE)->block.block_num)
```

이 block 을 위한 index 번호이다. 이 값들은 각 함수마다 유일하다는 것을 보장하지 않는데 – 사용중인 debugging output format 에 의존하기 때문이다.

```
#define BLOCK_FRAGMENT_ORIGIN(NODE) (BLOCK_CHECK (NODE)->block.fragment_origin)
#define BLOCK_FRAGMENT_CHAIN(NODE) (BLOCK_CHECK (NODE)->block.fragment_chain)
```

만약 block 재순서화 (reordering) 가 lexical block 을 접촉하지 않는 주소 범위로 나눠버린다면, 우리는 원래 block 의 복사본을 만들 것이다.

이것은 BLOCK_ABSTRACT_ORIGIN 로부터 논리적인 구분임을 알아라. 그러한 경우에, 우리는 많은 논리적 block 들로 (inlining 혹은 unrolling 을 통해) 사본을 뜬 하나의 source block 을 가지고 있고, 그러한 논리적 block 들은 그들 내부적으로 다른 물리적 변수들을 가지고 있다.

이 경우에, 우리는 여러 비-연속적인 주소 범위로 나눈 하나의 노리적 block 을 가지고 있다. 대부분의 debug format 들은 실제로 이 아이디어를 직관적으로 표현할수 없으며, 그래서 우리는 그들내부의 몇은 변수들로 다수의 논리적 block 을 생성함으로써 그것을 속인다. 하지만, 비-연속적 지역들을 지원하지 않는 것들에 대해서는, 이것들은 주소 범위의 집합(set) 에 따라 재건설한 원래의 논리 block 를 허락한다.

논리적 block 조각들의 하나는 ORIGIN 로 임의적으로 선택된다. 다른 조각들은 BLOCK_FRAGMENT_ORIGIN 을 통해 원래 형태를 가르킬 것이다; 원래 형태 자체는 이 pointer 를 null 로 가질 것이다. 조각들의 목록 은 origin 로 부터 BLOCK_FRAGMENT_CHAIN 를 통해 연결될 것이다.

3.2.12 tree_type 구조체

```
#define TYPE_UID(NODE) (TYPE_CHECK (NODE)->type.uid)
#define TYPE_SIZE(NODE) (TYPE_CHECK (NODE)->type.size)
#define TYPE_SIZE_UNIT(NODE) (TYPE_CHECK (NODE)->type.size_unit)
#define TYPE_MODE(NODE) (TYPE_CHECK (NODE)->type.mode)
#define TYPE_VALUES(NODE) (TYPE_CHECK (NODE)->type.values)
#define TYPE_DOMAIN(NODE) (TYPE_CHECK (NODE)->type.values)
#define TYPE_FIELDS(NODE) (TYPE_CHECK (NODE)->type.values)
#define TYPE_METHODS(NODE) (TYPE_CHECK (NODE)->type.maxval)
#define TYPE_VFIELD(NODE) (TYPE_CHECK (NODE)->type.minval)
#define TYPE_ARG_TYPES(NODE) (TYPE_CHECK (NODE)->type.values)
#define TYPE_METHOD_BASETYPE(NODE) (TYPE_CHECK (NODE)->type.maxval)
#define TYPE_OFFSET_BASETYPE(NODE) (TYPE_CHECK (NODE)->type.maxval)
#define TYPE_POINTER_TO(NODE) (TYPE_CHECK (NODE)->type.pointer_to)
#define TYPE_REFERENCE_TO(NODE) (TYPE_CHECK (NODE)->type.reference_to)
#define TYPE_MIN_VALUE(NODE) (TYPE_CHECK (NODE)->type.minval)
#define TYPE_MAX_VALUE(NODE) (TYPE_CHECK (NODE)->type.maxval)
#define TYPE_PRECISION(NODE) (TYPE_CHECK (NODE)->type.precision)
#define TYPE_SYMTAB_ADDRESS(NODE) (TYPE_CHECK (NODE)->type.symtab.address)
#define TYPE_SYMTAB_POINTER(NODE) (TYPE_CHECK (NODE)->type.symtab.pointer)
#define TYPE_NAME(NODE) (TYPE_CHECK (NODE)->type.name)
#define TYPE_NEXT_VARIANT(NODE) (TYPE_CHECK (NODE)->type.next_variant)
#define TYPE_MAIN_VARIANT(NODE) (TYPE_CHECK (NODE)->type.main_variant)
#define TYPE_CONTEXT(NODE) (TYPE_CHECK (NODE)->type.context)
#define TYPE_LANG_SPECIFIC(NODE) (TYPE_CHECK (NODE)->type.lang_specific)
```

이러한 field 들의 사용에 관한 문서는 tree.def 파일에서 볼수 있습니다. 여러 ...._TYPE tree code 들의 문서를 보십시오.

```
#define TYPE_DEBUG_REPRESENTATION_TYPE(NODE) (TYPE_CHECK (NODE)->type.values)
```

VECTOR_TYPE node 에서는, 이것은 debugging output 에 내보내야되는 다른 type 을 표현한다. 우리는 이것을 배열을 포함하는 구조체로써 vector 를 표현할 때 사용한다.

Indirect type 들은 어려움들을 나타내는데, 그들이 POINTER_TYPE/REFERENCE_TYPE node 들 (unbounded) 혹은 RECORD_TYPE node 들 (bounded) 로 표현될 수 있기 때문이다. Bounded 와 unbounded pointer 들은 논리적으로 동등하지만, 물리적으로는 다를 수 있다. main variant 의 간단한 비교로 type 들이 논리적으로 같은지를 말할 수 있다. 물리적 동일성에 대한 비교를 위해서는 이 술어를 사용하라.

```
#define TYPE_MAIN_VARIANTS_PHYSICALLY_EQUAL_P(TYPE1, TYPE2) \
  (TYPE_MAIN_VARIANT (TYPE1) == TYPE_MAIN_VARIANT (TYPE2)   \
   && TREE_CODE (TYPE1) == TREE_CODE (TYPE2))
```

Type 들은 같은 main variant 를 가지고 있고 같은 boundedness 를 가지고 있다.

```
#define TYPE_MAIN_PHYSICAL_VARIANT(TYPE)        \
  (BOUNDED_POINTER_TYPE_P (TYPE)            \
   ? build_qualified_type (TYPE, TYPE_QUAL_BOUNDED) \
   : TYPE_MAIN_VARIANT (TYPE))
```

Qualifier 를 가지고 있지 않은 type variant (예를 들면, main variant) 를 반환한다. 단, boundedness qualifier 가 보존되어 있다면 예외이다.

```
#define TYPE_BINFO(NODE) (TYPE_CHECK (NODE)->type.binfo)
```

그 자체의 기본 type 으로써, 이 type 에 관한 정보, type 들을 수집하기 위해. RECORD_TYPE 도 QUAL_UNION_TYPE, UNION_TYPE 도 아닌 type 들을 위한 language-dependent 방식에 사용된다.

```
#define TYPE_ALIAS_SET(NODE) (TYPE_CHECK (NODE)->type.alias_set)
```

이 type 을 위한 (언어-지정적인) typed-based alias 설정. TYPE_ALIAS_SET 들이 다른 object 들은 서로 alias 될 수 없다. 만약 TYPE_ALIAS_SET 가 -1 이면, 아직 아무런 alias set 이 이 type 에 할당되지 않은 것이다. 만약 TYPE_ALIAS_SET 가 0 이면, 이 type 의 object 들은 다른 type 의 object 들에 alias 될 수 있다.

```
#define TYPE_ALIAS_SET_KNOWN_P(NODE) (TYPE_CHECK (NODE)->type.alias_set != -1)
```

만약 이 type 을 위한 typed-based alias set 이 계산되어졌다면 0 이 아닌 값을 가짐.

```
#define TYPE_ATTRIBUTES(NODE) (TYPE_CHECK (NODE)->type.attributes)
```

이 type 에 반영할 attribute 들의 IDENTIFIER node 들의 TREE_LIST.

```
#define TYPE_ALIGN(NODE) (TYPE_CHECK (NODE)->type.align)
```

이 type 의 object 들을 위한 alignment 필요성. 값은 bit 들로 잰, int 이다.

```
#define TYPE_USER_ALIGN(NODE) (TYPE_CHECK (NODE)->type.user_align)
```

만약 이 type 을 위한 alignment 가 "aligned" attribute 에 의해 요철될 경우 1, 만약 그것이 이 type 의 기본값일 경우 0.

```
#define TYPE_ALIGN_UNIT(NODE) (TYPE_ALIGN (NODE) / BITS_PER_UNIT)
```

Byte 크기, NODE 를 위한 alignment.

```
#define TYPE_STUB_DECL(NODE) TREE_CHAIN (NODE)
```

만약 여러분의 언어가 type 들을 선언할 수 있도록 허락하고, 당신이 또한 그것 에 대한 debug info 를 원한다면 상응하는 TYPE_DECL node 들을 생성할 필요가 있다. 이러한 "stub" TYPE_DECL node 들은 이름을 가지고 있지 않으며, type node 로 간단히 가르키고 있다. TYPE_DECL node 로 다시 가르킬려면 type node 의 TYPE_STUB_DECL field 를 설정하면 된다. 이것은 두개의 node 들이 같은 type 을 나타내고 있음을 알려주는 debug routine 들을 허락하고 그래서 우리는 그들로부터 하나의 debug info record 를 얻을 수 있다.

```
#define TYPE_NO_FORCE_BLK(NODE) (TYPE_CHECK (NODE)->type.no_force_blk_flag)
```

RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE 에서 이것은 type 이 BLKmode 를 가지고 있고, 단지 그것의 크기를 위한 alignment 필요성을 누락시켰기 때문임을 의미한다.

```
#define TYPE_IS_SIZETYPE(NODE) \
  (INTEGER_TYPE_CHECK (NODE)->type.no_force_blk_flag)
```

INTEGER_TYPE 에서는 이것은 type 이 size 를 나타내고 있음을 의미한다. 우리는 이것을 타당성 검사와 다른 type 들에는 안전하지 못하는 최적화를 허락하는데 사용한다. C 에서의 'size_t' type 은 절대 이 flag 를 설정해서 는 안된다. 'size_t' type 은 'sizeof' 에 의해 반환되는 표현의 type 에 의해 발생되는 보통의 정수 type 을 위한 간단한 typedef 이다; 'size_t' 는 특별한 특성을 가지고 있지 않다. type 이 TYPE_IS_SIZETYPE 가 설정된 표현들은 항상 실제 size 들이다.

```
#define TYPE_RETURNS_STACK_DEPRESSED(NODE) \
  (FUNCTION_TYPE_CHECK (NODE)->type.no_force_blk_flag)
```

FUNCTION_TYPE 에서는, 함수가 낮아진 (depressed) stack pointer 와 함께 반환된을 지적한다.

```
#define TYPE_VOLATILE(NODE) (TYPE_CHECK (NODE)->common.volatile_flag)
```

Type 이 volatile 을 정수로써 간주할 경우에 값이 0 이 아님.

```
#define TYPE_READONLY(NODE) (TYPE_CHECK (NODE)->common.readonly_flag)
```

이 type 이 const-qualified 임을 의미한다.

```
#define TYPE_RESTRICT(NODE) (TYPE_CHECK (NODE)->type.restrict_flag)
```

만약 값이 0 이 아닐 경우, 이 type 은 C 부류의 단어로써 'restrict'-qualified 이다.

```
#define TYPE_BOUNDED(NODE) (TYPE_CHECK (NODE)->common.bounded_flag)
```

만약 값이 0 이 아닐 경우, 이 type 의 크기와 layout (혹은 FUNCTION_TYPE 혹은 METHOD_TYPE 의 경우에서 그것의 argument 들과/혹은 return 값 의 크기과 layout) 가 pointer bound 들의 존재로 인해 변경되었다.

```
#define TYPE_UNQUALIFIED   0x0
#define TYPE_QUAL_CONST    0x1
#define TYPE_QUAL_VOLATILE 0x2
#define TYPE_QUAL_RESTRICT 0x4
#define TYPE_QUAL_BOUNDED  0x8
```

각 type qualifier 을 위한 TYPE_QUAL 값이 있다. 그들은 type 을 위한 qualifier 들의 완전 집합을 형성하는데, bitwise-or 로 조합될 수 있다.

```
#define TYPE_QUALS(NODE)                    \
  ((TYPE_READONLY (NODE) * TYPE_QUAL_CONST)         \
   | (TYPE_VOLATILE (NODE) * TYPE_QUAL_VOLATILE)        \
   | (TYPE_RESTRICT (NODE) * TYPE_QUAL_RESTRICT)        \
   | (BOUNDED_INDIRECT_TYPE_P (NODE) * TYPE_QUAL_BOUNDED))
```

이 type 을 위한 type qualifier 들의 집합.

```
#define TREE_EXPR_QUALS(NODE)               \
  ((TREE_READONLY (NODE) * TYPE_QUAL_CONST)     \
   | (TREE_THIS_VOLATILE (NODE) * TYPE_QUAL_VOLATILE)   \
   | (TREE_BOUNDED (NODE) * TYPE_QUAL_BOUNDED))
```

expression node 을 위한 적당한 qualifier 들의 집합.

```
#define TREE_FUNC_QUALS(NODE)               \
  ((TREE_READONLY (NODE) * TYPE_QUAL_CONST)     \
   | (TREE_THIS_VOLATILE (NODE) * TYPE_QUAL_VOLATILE))
```

FUNCTION_DECL node 을 위한 적당한 qualifier 들의 집합.

```
#define TYPE_LANG_FLAG_0(NODE) (TYPE_CHECK (NODE)->type.lang_flag_0)
#define TYPE_LANG_FLAG_1(NODE) (TYPE_CHECK (NODE)->type.lang_flag_1)
#define TYPE_LANG_FLAG_2(NODE) (TYPE_CHECK (NODE)->type.lang_flag_2)
#define TYPE_LANG_FLAG_3(NODE) (TYPE_CHECK (NODE)->type.lang_flag_3)
#define TYPE_LANG_FLAG_4(NODE) (TYPE_CHECK (NODE)->type.lang_flag_4)
#define TYPE_LANG_FLAG_5(NODE) (TYPE_CHECK (NODE)->type.lang_flag_5)
#define TYPE_LANG_FLAG_6(NODE) (TYPE_CHECK (NODE)->type.lang_flag_6)
```

다음의 flag 들은 각 language front end 가 내부적으로 이용가능하다.

```
#define TYPE_STRING_FLAG(NODE) (TYPE_CHECK (NODE)->type.string_flag)
```

만약 ARRAY_TYPE 에서 설정된다면, 문자열 type 임 (char 의 배열로부터 문자열을 구분하는 언어를 위해) 을 나타낸다. 만약 SET_TYPE 에서 설정된다면, bitstring type 임을 나타낸다.

```
#define TYPE_ARRAY_MAX_SIZE(ARRAY_TYPE) \
  TYPE_MAX_VALUE (ARRAY_TYPE_CHECK (ARRAY_TYPE))
```

만약 NULL 이 아니면, 이것은 주어진 ARRAY_TYPE 의 object 의 크기 (byte 인) 의 upper bound 이다. 이것은 임시 할당을 허용한다.

```
#define TYPE_VECTOR_SUBPARTS(VECTOR_TYPE) \
  GET_MODE_NUNITS (VECTOR_TYPE_CHECK (VECTOR_TYPE)->type.mode)
```

VECTOR_TYPE 을 위해, 이것은 해당 vector 의 sub-part 들의 번호이다.

```
#define TYPE_NEEDS_CONSTRUCTING(NODE) \
  (TYPE_CHECK (NODE)->type.needs_constructing_flag)
```

이 type 의 object 들이 그들이 생성될 때 함수를 호출함으로써 반드시 초기화 되어야 함을 지시한다.

```
#define TYPE_TRANSPARENT_UNION(NODE)  \
  (UNION_TYPE_CHECK (NODE)->type.transparent_union_flag)
```

이 type (UNION_TYPE) 의 object 들이 동일한 첫번째 union 이 통과했던 같은 방법으로 반드시 통과해야 함을 지시한다.

```
#define TYPE_NONALIASED_COMPONENT(NODE) \
  (ARRAY_TYPE_CHECK (NODE)->type.transparent_union_flag)
```

ARRAY_TYPE 에 대해서, 해당 type 의 component 의 주소를 가지는 것이 허락 되지 않았음을 지시한다.

```
#define TYPE_PACKED(NODE) (TYPE_CHECK (NODE)->type.packed_flag)
```

이 type 의 object 들이 가능한 compact 한 방식으로 layout 되어야 함을 지시한다.

bounded pointer 혹은 bounded reference type (집합적으로 indirect type 들이라고 불리는) 은 type 의 세 pointer field 가 상응하는 unbounded POINTER_TYPE 혹은 REFERENCE_TYPE 인 세 pointer field 를 포함하는 RECORD_TYPE node 로써 나타내어진다. bounded indirect type 를 나타내는 RECORD_TYPE node 는 보통의 RECORD_TYPE node 와는 다른데, 그것의 TREE_TYPE 이 NULL 이 아니고, POINTER_TYPE 혹은 REFERENCE_TYPE node 를 가진 것 같은 pointed-to type 을 가지고 있다. bounded RECORD_TYPE node 들은 underlaying indirect types node 들의 variant 들과 나란히 같은 type variant chain 상에 저장된다. 그러한 chain 들의 main variant 는 항상 unbounded type 이다.

```
#define TYPE_BOUNDED_VALUE(TYPE) TYPE_FIELDS (TYPE)
#define TYPE_BOUNDED_BASE(TYPE) TREE_CHAIN (TYPE_BOUNDED_VALUE (TYPE))
#define TYPE_BOUNDED_EXTENT(TYPE) TREE_CHAIN (TYPE_BOUNDED_BASE (TYPE))
```

bounded-pointer type 의 field decls 로 접근한다.

```
#define TYPE_BOUNDED_SUBTYPE(TYPE) TREE_TYPE (TYPE_BOUNDED_VALUE (TYPE))
```

bounded-pointer type 의 simple-pointer subtype 에 접근한다.

```
#define TYPE_UNBOUNDED_VARIANT(TYPE) \
  (BOUNDED_POINTER_TYPE_P (TYPE) ? TYPE_BOUNDED_SUBTYPE (TYPE) : (TYPE))
```

Type 의 unbounded counterpart 를 찾거나, 그것이 이미 unbounded 라면 TYPE 을 반환한다.

```
#define TYPE_POINTER_DEPTH(TYPE) (TYPE_CHECK (TYPE)->type.pointer_depth)
```

이 field 는 두 bit 를 함유하고 있으며 0..3 범위의 값을 위해 존재한다.

depth=0 은 type 이 scalar 혹은 오직 depth=0 type 들만 포함하는 집합체 혹은, 그것의 return 값과 argument type 을 위해 오직 depth=0 type 들만 가지는 함수를 의미한다.

depth=1 은 type 이 depth=0 type 으로의 pointer 혹은, 오직 depth=0 와 depth=1 type 들만 포함하는 집합체, 혹은 그것의 return 값과 argument type 을 위해 오직 depth=0 와 depth=1 type 들만 가지는 함수를 의미한다.

depth=2 와 depth=3 의 의미들은 유도에 의해 분명해진다. Varargs function 들은 depth=3 이다. type 'va_list' 는 depth=3.이다.

type 의 pointer 깊이를 재는 목적은 자동적으로-생성된 bounded-pointer thunk 를 위한 함수의 적합성을 결정하기 위해서다. depth=0 function 들은 thunk 가 필요없다. depth=1 function 는 자동적인 thunk 에 적 합하다. depth 2 혹은 그 이상을 가진 함수는 automatic thunk 를 얻는데 너무 복잡하다.

함수 decl 들은 또한 pointer_depth field 고 있으므로, 또한 우리는 함수를 위한 실제 argument type 들을 고려할 수 있다.

```
#define TYPE_AMBIENT_BOUNDEDNESS(TYPE) \
  (FUNCTION_TYPE_CHECK (TYPE)->type.transparent_union_flag)
```

FUNCTION_TYPE node 에서 이 bit 는 time TYPE 이 생성었을 때 당시 default_pointer_boundedness 의 값을 저장하고 있다. 이것은 non-prototype function decls 와 varargs/stdarg list 들을 위한 함수의 argument 의 기본 boundedness 를 선택하는데 유용한다.

```
#define MAX_POINTER_DEPTH 2
#define VA_LIST_POINTER_DEPTH 3
```

설명 없음.

3.2.13 tree_type 구조체 : binfo

type inheritance 와 basetype 들에 관한 정보를 위한 연결재 매크로들을 정의합니다.

"basetype" 는 다른 type 내의 상속을 위한 data type 의 특정 사용을 의미한다. 각각의 그러한 basetype 사용은 그것을 설명하는 자신의 "binfo" 를 가지고 있다. Binfo object 는 TREE_VEC node 이다.

상속은 주어진 type 으로 할당된 binfo node 들로 나타난다. 예를 들면, 주어진 type 들 C 와 D 가 있고 D 가 C 에 의해 상속받을 경우, 3 binfo node 들이 할당될 것이다: C 의 binfo 특성을 설명하는 것, 비슷하게, D 를 위한 것, 그리고 C 를 위한 base type 으로써 D 의 binfo 특성을 설명하는 것이 것이다. 그래서 주어진 class C 로의 pointer 는 C 의 binfo 의 basetype 들을 살펴 봄으로써 C 를 위한 basetype 처럼 행동하는 D 의 binfo 로의 pointer 를 얻을 수 있다.

```
#define BINFO_TYPE(NODE) TREE_TYPE (NODE)
```

이 basetype 에 상속되어있는 실제 data type node.

```
#define BINFO_OFFSET(NODE) TREE_VEC_ELT ((NODE), 1)
#define TYPE_BINFO_OFFSET(NODE) BINFO_OFFSET (TYPE_BINFO (NODE))
#define BINFO_OFFSET_ZEROP(NODE) (integer_zerop (BINFO_OFFSET (NODE)))
```

이 basetype 이 그것을 포함하는 type 의 어디에서 나타나는지를 가르키는 offet.
BINFO_OFFSET slot 은 완료된 object 의 base 부터 이 'type' 측 에 할당된 object 부분의 base 까지의 (바이트 크기의) offset 을 가지고 있다. 다수의 상속이 있을 경우를 제외하곤 항상 0 이다.

```
#define BINFO_VTABLE(NODE) TREE_VEC_ELT ((NODE), 2)
#define TYPE_BINFO_VTABLE(NODE) BINFO_VTABLE (TYPE_BINFO (NODE))
```

이 basetype 을 따라는 virtual function table. Virtual function table 들은 run-time method dispatching 을 위한 메카니즘을 제공한다. virtual function table 의 entry 들은 언어-의존적인 부분이다.

```
#define BINFO_VIRTUALS(NODE) TREE_VEC_ELT ((NODE), 3)
#define TYPE_BINFO_VIRTUALS(NODE) BINFO_VIRTUALS (TYPE_BINFO (NODE))
```

virtual function table 내의 virtual function 들. 이것은 이 basetype 을 위해 virtual function table 을 생성하는 부분에서 초기의 접근으로 사용된 적이 있는 TREE_LIST 이다.

```
#define BINFO_BASETYPES(NODE) TREE_VEC_ELT ((NODE), 4)
#define TYPE_BINFO_BASETYPES(NODE) TREE_VEC_ELT (TYPE_BINFO (NODE), 4)
```

이 basetype 로 상속된 direct basetype 들을 위한 binfo 들의 vector.

만약 이 basetype 이 C 내로 상속된 type D 를 표현하고, D 의 basetype 이 E 와 F 라면, 그럼 이 vector 는 C 로의 E 와 F 의 상속에 대한 binfo 들을 포함한다.

??? 이것은 아마도 이 TREE_VEC (다른 TREE_VEC 을 사용하는 대신) 의 끝 부분에 단순히 base type 들을 할당함으로써 수행될 수 있다. 이것은 주어진 type 이 얼마나 많은 basetype 들을 가지고 있는지를 간산하게 계산할 수 있도록 할 것이다.

```
#define BINFO_N_BASETYPES(NODE) \
  (BINFO_BASETYPES (NODE) ? TREE_VEC_LENGTH (BINFO_BASETYPES (NODE)) : 0)
```

NODE 를 위한 basetype 들의 번호.

```
#define BINFO_BASETYPE(NODE,N) TREE_VEC_ELT (BINFO_BASETYPES (NODE), (N))
#define TYPE_BINFO_BASETYPE(NODE,N) \
  BINFO_TYPE (TREE_VEC_ELT (BINFO_BASETYPES (TYPE_BINFO (NODE)), (N)))
```

이 basetype 의 N 번째 basetype 을 얻기 위한 접근 매크로.

```
#define BINFO_VPTR_FIELD(NODE) TREE_VEC_ELT (NODE, 5)
```

virtual base class 를 나타내는 BINFO record 용으로, 예를 들면, TREE_VIA_VIRTUAL 가 설정된 것, 이 field 는 virtual base 위치 관련으로 도움을 준다. 실제 내용은 언어-독립적이다. 옛날 ABI 하에서, C++ front-end 는 내용의 FIELD_DECL 가 virtual base 로의 pointer 인 FIELD_DECL 를 사용하는데, 새로운 ABI 하에서 이 field 는 virtual base 의 offset 을 어디서 찾을 수 있는지에 대한 정보를 가지고있는 vtable 내의 offset 정보를 제공하는 INTEGER_CST 대신이다.

```
#define BINFO_SIZE(NODE) TREE_VEC_ELT (NODE, 6)
#define BINFO_SIZE_UNIT(NODE) TREE_VEC_ELT (NODE, 7)
#define TYPE_BINFO_SIZE(NODE) BINFO_SIZE (TYPE_BINFO (NODE))
#define TYPE_BINFO_SIZE_UNIT(NODE) BINFO_SIZE_UNIT (TYPE_BINFO (NODE))
```

이 type 의 base class subobject 의 크기. 하지만 현재 모든 frontend 들이 이러한 field 들을 위한 공간을 할당하지는 않는다.

```
#define BINFO_INHERITANCE_CHAIN(NODE) TREE_VEC_ELT ((NODE), 0)
```

상속의 사용을 나타내는 chain 을 생성하는데 사용되는 slot. 예를 들면, 만약 X 가 Y 로부터 유도되고 Y 가 Z 로부터 유도된다면 , 이 field 는 X 를 위한 binfo node 를 X 에서 Y 까지의 상속의 사용 을 나타내는 "X 의 Y" 를 위한 binfo node 로 연결하는데 사용될 수 있다. 비슷하게, "X 의 Y" 를 위한 binfo node 의 이 slot 은 Y 가 Z 로 부터 상속되어지는(X 의 상속 체계내에서) 것을 가르킬 수 있다. 이러한 형식으로, 어떤 것이 그들 스스로 binfo node 들을 사용하는 상속의 특정 사용법을 나타낼 수 있고 돌아다닐 수 있다. (Binfo node 들을 가르키기 위해 새로우 공간을 할당하는 대신 말이다.) 이것은 이 정보를 유지하는 언어-독립적인 front-end 들이 어떻게 필요하느냐에 달려있다.

3.2.14 tree_decl 구조체

declared name 들을 나타내고 있는 node 들을 위한 field 와 연결자 매크로를 정의합니다.

```
#define DECL_P(DECL)    (TREE_CODE_CLASS (TREE_CODE (DECL)) == 'd')
```

0 이 아닌 경우 DECL 은 decl 을 나타냅니다.

```
#define DECL_NAME(NODE) (DECL_CHECK (NODE)->decl.name)
```

이것은 사용자가 쓴 object 의 이름입니다. IDENTIFIER_NODE 를 가르킵니다.

```
#define DECL_ASSEMBLER_NAME(NODE)       \
  ((DECL_ASSEMBLER_NAME_SET_P (NODE)        \
    ? (void) 0                  \
    : (*lang_set_decl_assembler_name) (NODE)),  \
   DECL_CHECK (NODE)->decl.assembler_name)
```

어셈블러가 나중에 살필 해당 object 의 이름 (하지만 어떠한 해석이 ASM_OUTPUT_LABELREF 에 의해 만들어지기 전에). 이 값은 DECL_NAME 과 같은 수 있습니다. 값은 IDENTIFIER_NODE 입니다.

```
#define DECL_ASSEMBLER_NAME_SET_P(NODE) \
  (DECL_CHECK (NODE)->decl.assembler_name != NULL_TREE)
```

만약 NODE 를 위한 DECL_ASSEMBLER_NAME 가 설정되어 있다면 0 이 아닌 값을 되돌려 줍니다. 만약 0 이라면, NODE 는 아마 여전히 DECL_ASSEMBLER_NAME 를 가지고 있을 것이고 – 그것은 아직까지 설정되지 않았다.

```
#define SET_DECL_ASSEMBLER_NAME(NODE, NAME) \
  (DECL_CHECK (NODE)->decl.assembler_name = (NAME))
```

NODE 를 위한 DECL_ASSEMBLER_NAME 를 NAME 으로 설정합니다.

```
#define COPY_DECL_ASSEMBLER_NAME(DECL1, DECL2)              \
  (DECL_ASSEMBLER_NAME_SET_P (DECL1)                    \
   ? (void) SET_DECL_ASSEMBLER_NAME (DECL2,                 \
                                     DECL_ASSEMBLER_NAME (DECL1))   \
   : (void) 0)
```

DECL_ASSEMBLER_NAME 를 DECL1 에서 DECL2 로 복사한다. 만약 DECL1 의 DECL_ASSEMBLER_NAME 가 아직 설정되지 않았다면, 이 매크로를 사용하는 것은 설정될 DECL 의 DECL_ASSEMBLER_NAME 도 영향을 주지 않을 것이다. 다른 말로 말해서, 이 매크로를 사용하는 문잭은 다음과 같이 사용하는 거과는 다르다:

SET_DECL_ASSEMBLER_NAME(DECL2, DECL_ASSEMBLER_NAME (DECL1))

이것은 DECL1 로 DECL_ASSEMBLER_NAME 를 설정하도록 시도할 것이다.

```
#define DECL_SECTION_NAME(NODE) (DECL_CHECK (NODE)->decl.section_name)
```

section attribute 내에 section 이름을 기록합니다. decl_attributes 에서 make_function_rtl 과 make_decl_rtl 에 이름을 건네주는데 사용됩니다.

```
#define DECL_CONTEXT(NODE) (DECL_CHECK (NODE)->decl.context)
#define DECL_FIELD_CONTEXT(NODE) (FIELD_DECL_CHECK (NODE)->decl.context)
```

FIELD_DECL 들에서 사용되면 이것은 필드가 어떤 구성원인지를 가르키는 RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE 를 가집니다. VAR_DECL 와 PARM_DECL, FUNCTION_DECL, LABEL_DECL, CONST_DECL node 들에서 사용되면 이것은 포함하는 함수를 위한 FUNCTION_DECL 혹은 포함하는 type 을 위한 RECORD_TYPE 혹은 UNION_TYPE, 만약 주어진 decl 이 “file scope” 를 가지고 있다면 NULL_TREE 를 가르킬 것입니다.

```
#define DECL_ATTRIBUTES(NODE) (DECL_CHECK (NODE)->decl.attributes)
```

DECL 에서 이것은 attribute 들이 저장될 field 이다.

```
#define DECL_FIELD_OFFSET(NODE) (FIELD_DECL_CHECK (NODE)->decl.arguments)
```

FIELD_DECL 에서 이것은, 바이트로 계산한, field 위치인데, 이것은 구조체 의 시작에서 가장 가까운 bit 를 포함하는 바이트의 위치를 말한다.

```
#define DECL_FIELD_BIT_OFFSET(NODE) (FIELD_DECL_CHECK (NODE)->decl.u2.t)
```

FIELD_DECL 에서 이것은, 비트 크기로 나타낸, DECL_FIELD_OFFSET 부터 field 의 첫번째 bit 의 offset 이다.

```
#define DECL_BIT_FIELD_TYPE(NODE) (FIELD_DECL_CHECK (NODE)->decl.result)
```

FIELD_DECL 에서 이것은 field 가 bit-field 였었는지에 대한 여부와 그것을 원래부터 지정했었던 type 을 지시한다. TREE_TYPE 은 수정되었을 것이다. (finish_struct 에서)

```
#define DECL_ARGUMENTS(NODE) (DECL_CHECK (NODE)->decl.arguments)
```

FUNCTION_DECL 에서는 ...DECL node 들의 chain 입니다. VAR_DECL 와 PARM_DECL 사용할 language-specific 을 위한 argument slot 을 보존한다.

```
#define DECL_RESULT_FLD(NODE) (DECL_CHECK (NODE)->decl.result)
```

이 field 는 decl.result 내의 어떤 것을 참고하는데 사용되고 garbage collector 에 의한 사용시만 의미가 있다.

```
#define DECL_RESULT(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.result)
```

FUNCTION_DECL 에서 return 값을 위한 decl 을 가지고 있습니다.

```
#define DECL_ORIGINAL_TYPE(NODE) (TYPE_DECL_CHECK (NODE)->decl.result)
```

TYPE_DECL 에서 “original” type 을 가지고 있습니다. (TREE_TYPE 은 복사본을 가지고 있습니다.)

```
#define DECL_ARG_TYPE_AS_WRITTEN(NODE) (PARM_DECL_CHECK (NODE)->decl.result)
```

PARM_DECL 에서 쓰여진 type 을 가지고 있다. (아마도, 함수 혹은 배열)

```
#define DECL_INITIAL(NODE) (DECL_CHECK (NODE)->decl.initial)
```

FUNCTION_DECL 용 일때는 BINDING 들의 tree 를 가지고 있고,
VAR_DECL 용 일때는 초기값을 가지고 있고,
PARM_DECL 용 일때는 사용되지 않습니다 – parameter 들을 위한 기본 값은 PARM_DECL slot 이 아닌 function 의 type 에 encode 된다.

```
#define DECL_ARG_TYPE(NODE) (PARM_DECL_CHECK (NODE)->decl.initial)
```

PARM_DECL 에서는 프로그램에서 보였던 type 과 다를 수 있는 argument 를 전달하는데 사용되는 data type 을 기록한다.

```
#define DECL_QUALIFIER(NODE) (FIELD_DECL_CHECK (NODE)->decl.initial)
```

QUAL_UNION_TYPE 내의 FIELD_DECL 를 위해, 값이 0 이 아닐 경우 field 가 type 을 차지하고 있음을 지시하는 표현식을 기록하고 있다.

```
#define DECL_SOURCE_FILE(NODE) (DECL_CHECK (NODE)->decl.filename)
#define DECL_SOURCE_LINE(NODE) (DECL_CHECK (NODE)->decl.linenum)
```

다음의 두 field 는 선언이 원시 코드(source code)의 어떤 곳에서 있었는 지를 나타냅니다. 만약 선언(declaration)이 여러 곳에서 나타난다면 (예를 들면 C 함수는 처음에 선언(declared)되었다가 나중에 정의(defined)된다.) 이 정보는 정의를 나타내고 있다.

```
#define DECL_SIZE(NODE) (DECL_CHECK (NODE)->decl.size)
```

비트 크기로, 표현식으로써 datum 의 크기를 가지고 있다. 상수가 될 필요는 없다.

```
#define DECL_SIZE_UNIT(NODE) (DECL_CHECK (NODE)->decl.size_unit)
```

비슷하지만 바이트 크기를 가진다.

```
#define DECL_ALIGN(NODE) (DECL_CHECK (NODE)->decl.u1.a.align)
```

자료들에 요구되는 alignment 를 bit 수로 가지고 있다.

```
#define DECL_ALIGN_UNIT(NODE) (DECL_ALIGN (NODE) / BITS_PER_UNIT)
```

바이트 크기로, NODE 의 alignment.

```
#define DECL_OFFSET_ALIGN(NODE) \
  (((unsigned HOST_WIDE_INT)1) << FIELD_DECL_CHECK (NODE)->decl.u1.a.off_align)
```

FIELD_DECL 을 위해, off_align 는 항상 0 으로 알려져 있는 DECL_FIELD_OFFSET 의 low-order bit 들의 갯수 (번호) 를 가지고 있다. 그래서 DECL_OFFSET_ALIGN 는 DECL_FIELD_OFFSET 가 가진 alignment 를 반환한다.

```
#define SET_DECL_OFFSET_ALIGN(NODE, X) \
  (FIELD_DECL_CHECK (NODE)->decl.u1.a.off_align = exact_log2 ((X) & -(X)))
```

DECL_ALIGN(NODE) 의 X 의 곱임을 표시한다.

```
#define DECL_USER_ALIGN(NODE) (DECL_CHECK (NODE)->decl.user_align)
```

1 일 경우 이 type 의 alignment 는 “aligned” attribute 에 의해 요청되었습니다.
0 일 경우 이 type 을 위한 기본값입니다.

```
#define DECL_MODE(NODE) (DECL_CHECK (NODE)->decl.mode)
```

variable 혹은 field 의 declaration 과 일치하는 machine mode 를 가지고 있습니다. FIELD_DECL 을 제외하면 항상 TYPE_MODE (TREE_TYPE (decl)) 와 같습니다.

```
#define DECL_RTL(NODE)                  \
  (DECL_CHECK (NODE)->decl.rtl              \
   ? (NODE)->decl.rtl                   \
   : (make_decl_rtl (NODE, NULL), (NODE)->decl.rtl))
```

variable 혹은 function 의 값을 위한 RTL expression 를 가지고 있습니다. 만약 PROMOTED_MODE 가 정의되어 있다면, 이 표현식의 mode 는 아마도 DECL_MODE 와 같지 않을 것이다. 그러한 경우에, DECL_MODE 는, DECL_RTL 의 mode 가 데이타를 포함하는데 실질적으로 사용되는 mode 이지만, 변수의 data type 과 호응하는 mode 를 포함한다.

이 값은 라벨들과, static storage duration 을 가진 함수들, 변수들에 대해서는 느리게 평가될 수 있다.

```
#define SET_DECL_RTL(NODE, RTL) (DECL_CHECK (NODE)->decl.rtl = (RTL))
```

NODE 를 위한 DECL_RTL 를 RTL 로 설정합니다.

```
#define DECL_RTL_SET_P(NODE)  (DECL_CHECK (NODE)->decl.rtl != NULL)
```

만약 NODE 를 위한 DECL_RTL 이 이미 설정되어 있다면 0 이 아닌 값을 반환합니다.

```
#define COPY_DECL_RTL(NODE1, NODE2) \
  (DECL_CHECK (NODE2)->decl.rtl = DECL_CHECK (NODE1)->decl.rtl)
```

NODE 1 에서 NODE2 로 RTL 를 복사한다. 만약 RTL 이 NODE1 에서 설정되어 있지 않다면, NODE2 에 설정되지 않을 것이다; 이것이 게으른 copy 이다.

```
#define DECL_RTL_IF_SET(NODE) (DECL_RTL_SET_P (NODE) ? DECL_RTL (NODE) : NULL)
```

설정되어 있다면 NODE 를 위한 DECL_RTL 이 있고, NULL 이면 설정되어 있지 않음.

```
#define DECL_LIVE_RANGE_RTL(NODE) (DECL_CHECK (NODE)->decl.live_range_rtl)
```

변수가 가능한 다른 register 로 이동할 수 있는 가능한 범위의 모든 INSN_LIST 를 가지고 있다.

```
#define DECL_INCOMING_RTL(NODE) (PARM_DECL_CHECK (NODE)->decl.u2.r)
```

PARM_DECL 를 위해, Data 가 실제로 지나갈 수 있는 stack slot 혹은 register 를 위한 RTL 를 가지고 있다.

```
#define DECL_SAVED_INSNS(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.u2.f)
```

FUNCTION_DECL 를 위해 만약 그것이 inline 이라면 saved insn chain 를 가지고 있다.

```
#define DECL_FUNCTION_CODE(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.u1.f)
```

FUNCTION_DECL 를 위해 만약 그것이 built-in 이라면 이것은 built-in operation 이 어떤 것인지를 인식시켜 준다.

```
#define DECL_VINDEX(NODE) (DECL_CHECK (NODE)->decl.vindex)
```

DECL_VINDEX 는 두가지 다른 방식으로 FUNCTION_DECLS 를 위해 사용된다. FUNCTION_DECL 를 포함하고 있는 struct 가 layout 되기 전에, 이 FUNCTION_DECL 가 virtual function 으로써 FUNCTION_DECL 로 대체할 수 있는데 이를 base class 로써 DECL_VINDEX 가 FUNCTION_DECL 를 가르킬 수 있다. 해당 class 가 layout 되었을 때, 이 pointer 는 virtual function table 로 index 하는데 사용이 적절한 INTEGER_CST 로 변경될 수 있다.

```
#define DECL_FCONTEXT(NODE) (FIELD_DECL_CHECK (NODE)->decl.vindex)
```

FIELD_DECLS 를 위해, DECL_FCONTEXT 는 이 FIELD_DECL 가 정의되어 있는 *첫번째* baseclass 이다. 이 정보는 C++ 를 위한 vfield 와 vbase decl 들 에 관한 debugging 정보를 작성할 때 필요하다.

```
#define DECL_UID(NODE) (DECL_CHECK (NODE)->decl.uid)
```

모든 ...._DECL node 는 유일한 번호를 가지고 있습니다.

```
#define DECL_ABSTRACT_ORIGIN(NODE) (DECL_CHECK (NODE)->decl.abstract_origin)
```

...._DECL node 의 어느 분류를 위해, 이것은 이 decl 의 대신인 원래 (abstract) decl node 를 가르키거나 혹은 이 decl 기 몇몇 다른 decl 의 대신이 아님을 가르키는 NULL 을 가진다. 예를 들면, Inline 함수의 nested 선언에서 이것은 정의 (definition) 을 뒤로 가르킨다.

```
#define DECL_ORIGIN(NODE) \
  (DECL_ABSTRACT_ORIGIN (NODE) ? DECL_ABSTRACT_ORIGIN (NODE) : (NODE))
```

DECL_ABSTRACT_ORIGIN 와 비슷하지만 만약 abstract origin 이 없다면 NODE 를 반환한다. 이것은 DECL_ABSTRACT_ORIGIN 가 설정되어 있을 때 쓸모있다.

```
#define DECL_FROM_INLINE(NODE) (DECL_ABSTRACT_ORIGIN (NODE) != NULL_TREE \
                && DECL_ABSTRACT_ORIGIN (NODE) != (NODE))
```

...._DECL node 의 어느 분류에 대해 값이 0 이 아님은 이 decl node 가 inline function 로부터 몇몇 원래의 (abstract) decl 의 inline instance 를 나타냄 을 의미한다; 몇몇 다른 변수를 덮어두는 것에 대한 어떠한 경고를 잠재운다. FUNCTION_DECL node 들은 그들 자신을 설정하는 그들의 abstract origin 를 또한 가질 수 있다.

```
#define DECL_IGNORED_P(NODE) (DECL_CHECK (NODE)->decl.ignored_flag)
```

만약 _DECL 가 이 decl 의 이름이 symbolic debug 목적들을 위해 무시되어야 함을 의미한다면 값이 0 이 아님.

```
#define DECL_ABSTRACT(NODE) (DECL_CHECK (NODE)->decl.abstract_flag)
```

주어진 ...._DECL node 에 대해 값이 0 이 아님은 이 node 가 주어진 선언 (예를 들면, inline 함수의 원래 선언에서) 의 "abstract instance" 를 나타냄을 의미한다. symbolic debugging 정보를 생성할 때, 우리는 "abstract instances" 로 마킹된 node 들에 대해 어떠한 address information 를 생성하려고 시도해서는 안되는데, 왜냐하면 우리는 실제로 그러한 instance 들을 위해 어떠한 code 나 어떠한 data 공간을 할당하지 않기 때문이다.

```
#define DECL_IN_SYSTEM_HEADER(NODE) \
  (DECL_CHECK (NODE)->decl.in_system_header_flag)
```

0 이 아닐 경우 _DECL 는 "이 decl 은 단지 사용되지 않았기 때문에 어떠한 경고도 생성되어서는 않된다." 는 의미를 가지고 있습니다.

```
#define DECL_COMMON(NODE) (DECL_CHECK (NODE)->decl.common_flag)
```

주어진 ..._DECL node 가 가능한 .common 에 넣을려고 한다면 0 이 아닌 값을 가집니다. 만약 DECL_INITIAL 이 주어졌고 그것이 error_mark_node 가 아니라면 decl 은 .common 에 놓을 수가 없습니다.

```
#define DECL_LANG_SPECIFIC(NODE) (DECL_CHECK (NODE)->decl.lang_specific)
```

Language-specific decl 정보.

```
#define DECL_EXTERNAL(NODE) (DECL_CHECK (NODE)->decl.external_flag)
```

VAR_DECL 혹은 FUNCTION_DECL 에서 0 이 아닐 경우 external reference 를 의미 : storage 를 할당하지 않고 다른 곳의 정의를 참고합니다.

```
#define DEFAULT_INIT_PRIORITY 65535
#define MAX_INIT_PRIORITY 65535
#define MAX_RESERVED_INIT_PRIORITY 100
```

RECORD_TYPE 를 위한 VAR_DECL 에서, non-init_priority 초기화를 위한 숫자를 설정한다.

```
#define TYPE_DECL_SUPPRESS_DEBUG(NODE) \
  (TYPE_DECL_CHECK (NODE)->decl.external_flag)
```

TYPE_DECL 에서 값이 0 이 아님은 이 type 에 관한 상세 정보가 stabs 내에 덤프되지 않았음 을 의미한다. 대신에 그것은 name 들의 cross reference ('x') 를 생설할 것이다. 이것은 DECL_EXTERNAL 와 같은 flag 를 사용한다.

```
#define DECL_REGISTER(NODE) (DECL_CHECK (NODE)->decl.regdecl_flag)
```

VAR_DECL 와 PARM_DECL node 들에서 0 이 아닐 경우 'register' 로 선언되었음을 의미합니다.

```
#define DECL_ERROR_ISSUED(NODE) (LABEL_DECL_CHECK (NODE)->decl.regdecl_flag)
```

LABEL_DECL node 들에서, 값이 0 이 아님은 binding contour 와 같은 jump 에 관한 오류 메세지가 이 label 에 대해 출력되었음을 의미한다.

```
#define DECL_PACKED(NODE) (FIELD_DECL_CHECK (NODE)->decl.regdecl_flag)
```

FIELD_DECL 에서는 이 field 가 bit-pack 되어야 함을 가르킨다.

```
#define DECL_NO_STATIC_CHAIN(NODE) \
  (FUNCTION_DECL_CHECK (NODE)->decl.regdecl_flag)
```

값이 0 이 아닌 DECL_CONTEXT 를 가진 FUNCTION_DECL 에서는 static chain 이 필요하지 않음을 가르킨다.

```
#define DECL_NONLOCAL(NODE) (DECL_CHECK (NODE)->decl.nonlocal_flag)
```

....DECL 에서의 값이 0 이 아님은 이 변수가 nested function 에 의해 ref 됨을 의미한다. VAR_DECL node 들, PARM_DECL node 들, FUNCTION_DECL node 들 용이다.

LABEL_DECL node 용으로 만약 label 로의 nonlocal goto 들이 허락되지 않는다면 값이 0 이 아니다.

또한 몇몇 언어들에서 변수들, 기타 class instance 변수들과 같이 보통 lexical 범위 밖에 있는 것들에서 설정한다.

```
#define DECL_INLINE(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.inline_flag)
```

FUNCTION_DECL 에서 값이 0 이 아님은 이 함수가 그것이 호출된 것으로 대체될 수 있음을 의미한다.

```
#define DECL_UNINLINABLE(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.uninlinable)
```

FUNCTION_DECL 에서 만약 함수가 inline 될 수 없다면 값이 0 이 아님.

```
#define DECL_SAVED_TREE(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.saved_tree)
```

FUNCTION_DECL 에서, 전체 함수의 body 의 saved representation.
보통 COMPOUND_STMT 인데, 하지만 C++ 에서 이것은 RETURN_INIT, CTOR_INITIALIZER, 혹은 TRY_BLOCK 이다.

```
#define DECL_INLINED_FNS(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.inlined_fns)
```

이 함수의 body 로 inline 된 FUNCTION_DECL 들의 목록.

```
#define DECL_BUILT_IN_NONANSI(NODE) \
  (FUNCTION_DECL_CHECK (NODE)->common.unsigned_flag)
```

FUNCTION_DECL 에서 값이 0 이 아님은 이것이 built-in 함수임을 나타내며, 이 함수는 ansi C 에서 지정되지 않았으며 사용자들이 어떠한 무슨 목적을 위해서 재정의하는 것을 허락했다고 가정한다.

```
#define DECL_IS_MALLOC(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.malloc_flag)
```

FUNCTION_DECL 에서 값이 0 이 아님은 이 함수가 반드시 alias 를 하는 것이 아닌 pointer 를 반환하는 것을 의미하는 malloc 와 같이 취급되어야 함을 의미한다.

```
#define DECL_IS_PURE(NODE) (FUNCTION_DECL_CHECK (NODE)->decl.pure_flag)
```

FUNCTION_DECL 에서 값이 0 이 아님은 이 함수가 반드시 “pure” 함수 (const function 과 비슷하지만, global memory 를 읽는다.) 와 같이 취급되어야 함을 의미한다.

```
#define DECL_BIT_FIELD(NODE) (FIELD_DECL_CHECK (NODE)->decl.bit_field_flag)
```

FIELD_DECL 에서 값이 0 이 아님은 그것은 bit field 이며 반드시 특별하게 접근 되어야 함을 의미한다.

```
#define DECL_TOO_LATE(NODE) (LABEL_DECL_CHECK (NODE)->decl.bit_field_flag)
```

LABEL_DECL 에서 값이 0 이 아님은 label 이 stack level 가 저장되어 있고, 이제 막 빠져나온 binding contour 내에 정의되어 있음을 의미한다.

```
#define DECL_IN_TEXT_SECTION(NODE) (VAR_DECL_CHECK (NODE)->decl.bit_field_flag)
```

static 인 VAR_DECL 에서, 만약 공간 (space) 이 text section 내에 있을 경우 값이 0 이 아님.

```
#define DECL_BUILT_IN(NODE) (DECL_BUILT_IN_CLASS (NODE) != NOT_BUILT_IN)
```

FUNCTION_DECL 에서 값이 0 이 아님은 built in function 임을 의미.

```
#define DECL_BUILT_IN_CLASS(NODE) \
   (FUNCTION_DECL_CHECK (NODE)->decl.built_in_class)
```

builtin function 에서, 컴파일러의 어떤 부분에 그것이 정의되었는지를 가르켜준다.

```
#define DECL_VIRTUAL_P(NODE) (DECL_CHECK (NODE)->decl.virtual_flag)
```

변수가 vtable 임을 가르키기 위해 VAR_DECL 들에서 사용된다. vtable pointer 들을 위해 FIELD_DECL 들에서 사용된다. 함수가 virtual 임을 가르키기 위해 FUNCTION_DECL 에서 사용된다.

```
#define DECL_DEFER_OUTPUT(NODE) (DECL_CHECK (NODE)->decl.defer_output)
```

이 DECL 의 linkage status 가 아직 알려지지 않아서 현재로서는 output 해서는 안된다는 사실을 가르킬 때 사용됩니다.

```
#define DECL_TRANSPARENT_UNION(NODE) \
  (PARM_DECL_CHECK (NODE)->decl.transparent_union)
```

PARM_DECL 들의 type 은 argument 가 반드시 첫번째 union 동일 부분이 지나갔던 것과 똑같은 방식으로 지나가야 함을 가르키는 union 들이다. 이 부분은 PARM_DECL 들에서 사용된다.

```
#define DECL_STATIC_CONSTRUCTOR(NODE) \
  (FUNCTION_DECL_CHECK (NODE)->decl.static_ctor_flag)
#define DECL_STATIC_DESTRUCTOR(NODE) \
(FUNCTION_DECL_CHECK (NODE)->decl.static_dtor_flag)
```

그것들이 실행의 처음부터 끝까지 자동적으로 반드시 수행되어야 한가는 것을 가르키며 FUNCTION_DECL 들에서 사용된다.

```
#define DECL_ARTIFICIAL(NODE) (DECL_CHECK (NODE)->decl.artificial_flag)
```

이 DECL 이 compiler-generated entity 를 표현한다는 것을 가르키는데 사용된다.

```
#define DECL_WEAK(NODE) (DECL_CHECK (NODE)->decl.weak_flag)
```

이 DECL 은 weak linkage 를 가지고 있음을 지시하는데 사용됩니다.

```
#define DECL_ONE_ONLY(NODE) (DECL_CHECK (NODE)->decl.transparent_union)
```

다수의 translation unit 들내의 이 DECL 의 복사본들이 반드시 합병되어햐 함을 가르키는 TREE_PUBLIC decl 들내에서 사용된다.

```
#define DECL_COMDAT(NODE) (DECL_CHECK (NODE)->decl.comdat_flag)
```

심지어 그것이 TREE_PUBLIC 라도, 만약 그것이 이 translation unit 내에 필요하지 않는다면 그것을 올려놓을 필요가 없음을 가르키는 DECL 내에서 사용된다. 이것과 같은 entity 들은 translation unit 들 (weak entity 들 과 같은) 사이로 공유되어야 하지만 그들을 필요로 하는 어느 translation unit 에 의해 생성되는 것이 보장되어야 하고 그러기 때문에 그들이 필요 하지 않는 어떠한 곳에 올려놓을 필요는 없다. DECL_COMDAT 는 단지 back-end 를 위한 힌트이다; DECL_COMDAT 인 어떤 것을 올려 놓는데에 있어서 어떠한 해악도 생기지 않으며, 걸리적 거리는 것도 생기지 않음을 확신시키기 위해 이 flag 를 설정하는 것은 front-end 들에 책임이 있다.

```
#define DECL_NO_INSTRUMENT_FUNCTION_ENTRY_EXIT(NODE) \
  (FUNCTION_DECL_CHECK (NODE)->decl.no_instrument_function_entry_exit)
```

함수 entry 와 exit 는 반드시 루틴들을 지원하기 위해 호출 (call) 들을 가져야 함을 지시하며 FUNCTION_DECL 내에서 사용된다.

```
#define DECL_NO_LIMIT_STACK(NODE) \
  (FUNCTION_DECL_CHECK (NODE)->decl.no_limit_stack)
```

limit-stack-* 가 반드시 이 함수내에서는 불능으로 되어야 함을 가르키며 FUNCTION_DECL 들내에서 사용된다.

```
#define DECL_LANG_FLAG_0(NODE) (DECL_CHECK (NODE)->decl.lang_flag_0)
#define DECL_LANG_FLAG_1(NODE) (DECL_CHECK (NODE)->decl.lang_flag_1)
#define DECL_LANG_FLAG_2(NODE) (DECL_CHECK (NODE)->decl.lang_flag_2)
#define DECL_LANG_FLAG_3(NODE) (DECL_CHECK (NODE)->decl.lang_flag_3)
#define DECL_LANG_FLAG_4(NODE) (DECL_CHECK (NODE)->decl.lang_flag_4)
#define DECL_LANG_FLAG_5(NODE) (DECL_CHECK (NODE)->decl.lang_flag_5)
#define DECL_LANG_FLAG_6(NODE) (DECL_CHECK (NODE)->decl.lang_flag_6)
#define DECL_LANG_FLAG_7(NODE) (DECL_CHECK (NODE)->decl.lang_flag_7)
```

language-specific 사용을 위한 추가적인 flag 들.

```
#define DECL_NON_ADDR_CONST_P(NODE) (DECL_CHECK (NODE)->decl.non_addr_const_p)
```

이 DECL 로의 pointer 가 address constant 로써 취급될 수 없음을 가르키는데 사용된다.

```
#define DECL_NONADDRESSABLE_P(NODE) \
  (FIELD_DECL_CHECK (NODE)->decl.non_addressable)
```

이 component 의 address 를 형성할 수 없음을 가르키기 위해 FIELD_DECL 내에서 사용된다.

```
#define DECL_POINTER_ALIAS_SET(NODE) \
  (DECL_CHECK (NODE)->decl.pointer_alias_set)
```

반드시 pointer (혹은 reference) type 을 가져야 하는 이 특정 FIELD_DECL, PARM_DECL, 혹은 VAR_DECL 에 의해 가르켜져야 하는 메모리를 위한 alias set 를 지정하는데 사용된다.

```
#define DECL_POINTER_ALIAS_SET_KNOWN_P(NODE) \
  (DECL_POINTER_ALIAS_SET (NODE) != - 1)
```

만약 alias set 이 이 선언으로 할당되어 졌다면 값이 0 이 아님.

```
#define DECL_POINTER_DEPTH(DECL) (DECL_CHECK (DECL)->decl.pointer_depth)
```

pointer_depth field 는 범위가 0..3 까지의 값을 위한 두개의 bit 를 함유한다. 값이 보통 decl 의 type node 의 TYPE_POINTER_DEPTH 와 같지만 함수들에 대해서는 아마 더 클 것이다. 예를 들면, 이것은 type void* (depth=1) 의 parameter 를 받아들이는 함수가 선언되었을 때 일거날 수 있지만 실제로는 type foo** (depth=2) 의 argument 와 함께 호출될 것이다. 함수 type 은 형식적인 parameter 의 depth 를 얻을 것이지만, 함수 decl 은 실제 argument 의 depth 를 얻을 것이다.

제 4 절 TREE code

4.1 읽는 방법

아래의 나열은 어떤 순서로 나열되어 있는데,첫번째 인자는 tree code 의 이름을 나타내며, 두번째 인자는 tree code 의 분류를 나타내고, 마지막 세번째 인자는 현재 tree code 에 할당할 argument slot 들의 갯수를 나타낸다. 두번째 인자에 사용될 분류 기준은 아래와 같다.

x - 예외 코드일 경우. (적당한 분류를 찾지 못한 것들에 대해)

t - Type object code 일 경우

b - 어휘적 (lexical) block 용.

c - 상수들을 위한 code 들용.

d - 선언들 (또한 변수 참조들로써 제공되는 것들) 을 위한 code 용.

r - 저장소로의 참조 (reference) 들을 위한 code 용.

< - 비교 표현식들을 위한 code 들 용.

1 - Unary arithmetic 표현식을 위한 code 들용.

2 - Binary arithmetic 표현식을 위한 code 들용.

s - 선천적으로 부과적 영향을 가지는 표현식들을 위한 code 용.

e - 표현식의 다른 종류들을 위한 code 용.

그리고 각 하나의 code 마다 각각의 쓰임새에 대한 특정 코드를 포함하고 있는데, 이것은 해당 code 가 어떻게 사용해야 할지를 말해준다고 할 수 있다. 아래와 같은 코드를 가르키는데...

```
<tree_list 0x402316a4
    purpose <identifier_node 0x40178b80 format>
    value <tree_list 0x40180104
        value <identifier_node 0x40178a80 printf public
            global <function_decl 0x401855b0 printf>>
        chain <tree_list 0x4018008c
            value <integer_cst 0x40179e60 constant 1>
            chain <tree_list 0x40180028
                value <integer_cst 0x40179e40 constant 0>>>>>
```

이 정보는 $prefix/gcc/print-tree.c 파일에 선언되어 있는 함수 debug_tree () 를 사용하여 출력된 결과이다. 이 결과물로 인해서 문서의 양이 상대적으로 많이 늘어나버렸다.

4.2 tree code 들

이제 tree 는 어떤 code 들로 구성되어 있는지 하나하나 살펴보도록 하자. 아래의 code 들은 $prefix/gcc/tree.def 파일에 선언되어 있다. 또한 각각에 대한 자세한 설명 또한 포함하고 있다.

ERROR_MARK x 0

> 어떠한 오류가 있는 construct 는 이 type 의 node 로 해석된다. 이 node 의 type 은 하나의 오류에 대해 여러 오류 메세지들을 출력하는 것을 피하기 위해서 나중에 parsing 활동을 하기 때문에 모둔 문맥들에서 불평없이 받아진다. 다음 node 들의 field 들은 TREE_CODE 를 제외하고는 사용되지 않는다.

IDENTIFIER_NODE x -1

> 이름을 표현하는데 사용됩니다. (decl node 의 DECL_NAME 에서와 같이) 내부적으로 이것은 STRING_CST node 처럼 보입니다. There is only one IDENTIFIER_NODE ever made for any particular name. 이것을 얻기 위해서는 'get_identifier' 를 사용합니다. (처음일 경우 그것을 새로이 생성합니다.)

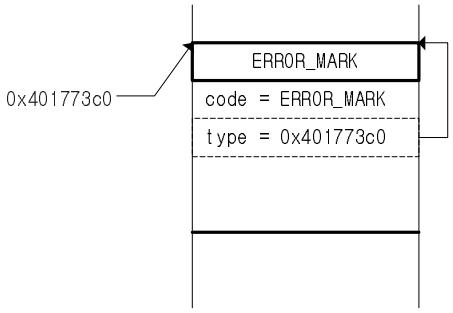

그림 1: 할당 완료된 ERROR_MARK

```
<identifier_node 0x40178b80 format>
<identifier_node 0x40178a80 printf public
    global <function_decl 0x401855b0 printf>>
```

TREE_LIST x 2

TREE_VALUE 와 TREE_PURPOSE field 들을 가지고 있습니다.
그러한 node 들은 TREE_CHAIN 필드를 통해 연결되어 있으므로해서 list 를 구성합니다. List 의 element 들은 TREE_VALUE field 들속에서 살고 있다. 그리고 TREE_PURPOSE field 는 경우에 따라 Lisp association list 의 효과 를 얻기 위해 역시 사용된다.

```
<tree_list 0x40164974 value <void_type 0x4017a770 void>>
<tree_list 0x402316a4
    purpose <identifier_node 0x40178b80 format>
    value <tree_list 0x40180104
        value <identifier_node 0x40178a80 printf public
            global <function_decl 0x401855b0 printf>>
        chain <tree_list 0x4018008c
            value <integer_cst 0x40179e60 constant 1>
            chain <tree_list 0x40180028
                value <integer_cst 0x40179e40 constant 0>>>>>
<tree_list 0x4022ae74 value <pointer_type 0x40229930>
    chain <tree_list 0x4022ae88 value <void_type 0x4017a770 void>>>
```

TREE_VEC x 2

이 node 들은 tree node 들의 배열을 포함하고 있습니다.

C 용 예제가 없습니다. 비록 TREE_VEC node 는 생성되나 예제로 보일 만한 것이 존재하지 않습니다.

BLOCK b 0

Symbol binding block. Tree 형식으로 정리되어 있으며, BLOCK_SUBBLOCKS field 는 BLOCK_CHAIN field 를 통해 연결된 subblock 들의 chain 을 포함하고 있다.
BLOCK_SUPERCONTEXT 는 부모 block 을 가르킨다.

함수의 가장 바깥쪽 scope 를 나타내는 block 에 대해, 그것은 FUNCTION_DECL node 를 가르킨다.

BLOCK_VARS 는 decl node 들의 chain 을 가르킨다.
BLOCK_TYPE_TAGS 는 그들 자신의 이름들을 가지고 있는 type 들의 chain 을 가르킨다.
BLOCK_CHAIN 는 같은 레벨에서의 다음 BLOCK 을 가르킨다.
BLOCK_ABSTRACT_ORIGIN 는 이 block 의 어떤것의 instance 인 원래 (추상적인) tree node 를 가르키며, 그렇지 않을 경우 이 block 이 어떤 다른 것의 instance 가 아님을 말하는 NULL 을 갖는다. NULL 이 아닐 경우, 값은 다른 BLOCK node 을 가르키거나 FUNCTION_DECL node 을 가르킬 수 있다. (예를 들면, 함수의 특정 inlining 의 가장 바깥쪽 scope 를 나타내는 block 의 경우)
BLOCK_ABSTRACT 는 만약 block 이 그것의 abstract instance (예를 들면, inline 함수의 abstract instance 내에 nested 한 것) 을 표현하는 것일면 0 이 아닌 값을 가진다.
TREE_ASM_WRITTEN 는 만약 block 이 생성된 어셈블리내에서 실제로 참조되었다면 0 이 아닌 값을 가진다.

```
<block 0x4030bf80 used
   subblocks <block 0x4030bf40 used
       vars <var_decl 0x4030fcb0 p
           type <pointer_type 0x4030f380
               type <pointer_type 0x4030f1c0 func_ptr
                   type <function_type 0x4017daf0
                       type <void_type 0x4017a770 void VOID
                           align 8 symtab 0 alias set -1
                           pointer_to_this <pointer_type 0x4017a7e0>>
                       DI
                       size <integer_cst 0x401792c0 constant 64>
                       unit size <integer_cst 0x401794e0 constant 8>
                       align 64 symtab 0 alias set -1
                       arg-types <tree_list 0x40164974
                           value <void_type 0x4017a770 void>>
                       pointer_to_this <pointer_type 0x402b53f0>>
                   unsigned SI
                   size <integer_cst 0x40179540 constant 32>
                   unit size <integer_cst 0x401795a0 constant 4>
                   align 32 symtab 0 alias set -1
                   pointer_to_this <pointer_type 0x4030f380>>
               unsigned SI size <integer_cst 0x40179540 32>
               unit size <integer_cst 0x401795a0 4>
               align 32 symtab 0 alias set -1>
           unsigned used common SI file crtstuff.c line 482
           size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
           align 32 context <function_decl 0x4030fbd0
               __do_global_ctors_aux>>
           supercontext <block 0x4030bf80>>>
<block 0x4030bf40 used
   vars <var_decl 0x4030fcb0 p
       type <pointer_type 0x4030f380
           type <pointer_type 0x4030f1c0 func_ptr
               type <function_type 0x4017daf0
                   type <void_type 0x4017a770 void VOID
                       align 8 symtab 0 alias set -1
                       pointer_to_this <pointer_type 0x4017a7e0>>
                   DI
                   size <integer_cst 0x401792c0 constant 64>
```

```
                    unit size <integer_cst 0x401794e0 constant 8>
                    align 64 symtab 0 alias set -1
                    arg-types <tree_list 0x40164974
                        value <void_type 0x4017a770 void>>
                    pointer_to_this <pointer_type 0x402b53f0>>
                unsigned SI
                size <integer_cst 0x40179540 constant 32>
                unit size <integer_cst 0x401795a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x4030f380>>
            unsigned SI size <integer_cst 0x40179540 32>
            unit size <integer_cst 0x401795a0 4>
            align 32 symtab 0 alias set -1>
        unsigned used common SI file crtstuff.c line 482
        size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
        align 32 context <function_decl 0x4030fbd0 __do_global_ctors_aux>>>
    <block 0x4026ed00 used
       subblocks <block 0x4026ecc0 used
           vars <var_decl 0x4028b7e0 __retval
               type <pointer_type 0x4017ac40
                   type <integer_type 0x40166230 char QI
                       size <integer_cst 0x40176da0 constant 8>
                       unit size <integer_cst 0x40176dc0 constant 1>
                       align 8 symtab 0 alias set -1 precision 8
                       min <integer_cst 0x40176e60 -128>
                       max <integer_cst 0x40176e80 127>
                       pointer_to_this <pointer_type 0x4017ac40>>
                   unsigned SI
                   size <integer_cst 0x40179540 constant 32>
                   unit size <integer_cst 0x401795a0 constant 4>
                   align 32 symtab 0 alias set -1
                   pointer_to_this <pointer_type 0x4022e9a0>>
               unsigned used in_system_header common regdecl SI
               file /usr/include/bits/string2.h line 1175
               size <integer_cst 0x40179540 32>
               unit size <integer_cst 0x401795a0 4>
               align 32 context <function_decl 0x4028b460 __strsep_3c>
               initial <indirect_ref 0x4028c2f8>>
               supercontext <block 0x4026ed00>
           subblocks <block 0x4026ec80 used
               vars <var_decl 0x4028b850 __cp
                   type <pointer_type 0x4017ac40>
                   unsigned used in_system_header common regdecl SI
                   file /usr/include/bits/string2.h line 1178
                   size <integer_cst 0x40179540 32>
                   unit size <integer_cst 0x401795a0 4>
                   align 32
                   context <function_decl 0x4028b460 __strsep_3c>
                   initial <var_decl 0x4028b7e0 __retval>>
                   supercontext <block 0x4026ecc0>>>>
    <block 0x4026ec80 used
       subblocks <block 0x4026ec40 used
```

```
vars <var_decl 0x4028b7e0 __retval
    type <pointer_type 0x4017ac40
        type <integer_type 0x40166230 char QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017ac40>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x4022e9a0>>
    unsigned used in_system_header common regdecl SI
    file /usr/include/bits/string2.h line 1175
    size <integer_cst 0x40179540 32>
    unit size <integer_cst 0x401795a0 4>
    align 32 context <function_decl 0x4028b460 __strsep_3c>
    initial <indirect_ref 0x4028c2f8>> supercontext <block 0x4026ec80>
subblocks <block 0x4026ec00 used
    vars <var_decl 0x4028b850 __cp type <pointer_type 0x4017ac40>
        unsigned used in_system_header common regdecl SI
        file /usr/include/bits/string2.h line 1178
        size <integer_cst 0x40179540 32>
        unit size <integer_cst 0x401795a0 4>
        align 32
        context <function_decl 0x4028b460 __strsep_3c>
        initial <var_decl 0x4028b7e0 __retval>>
        supercontext <block 0x4026ec40>>>>
```

각 data type 은 code 가 아래의 것들 중 하나로 표현되어집니다.
Data type 을 나타내는 각 node 는 bit 로 size 를 표현한 tree 를 포함하는 component TYPE_SIZE 를 가지고 있다. TYPE_MODE 는 이 type 의 값을 위한 machine mode 을 포함한다.
TYPE_POINTER_TO field 는 이 type 으로의 pointer 를 위한 type 을 포함 하거나 그러한 것이 아직 생성되지 않았다면 0 을 가진다.
TYPE_NEXT_VARIANT field 는 "const" 와 "volatile" 과 같은 type modifier 들로 만들어진 변수들을 type 으로 함께 묶는데 사용된다.
TYPE_MAIN_VARIANT field 는, 다음 chain 의 아무 요소 안의, chain 의 시작부분을 가르킨다.
TYPE_NONCOPIED_PARTS 는 이 type 의 object 의 어떤 부분이 assignment 로 절대 복사되지 말아야 할지를 지정하는 list 이다. 각각의 TREE_VALUE 는 복사되어서는 안되는 FIELD_DECL 이다. TREE_PURPOSE 는 이 type 의 object 가 INIT_EXPR 를 통해 초기화되어질 때 해당 field 를 위한 초기값이다. 특별한 값이 요구되어지지 않는다면 값은 NULL 일 것이다. assignment 의 right-hand side 가 완료된 object (아마도, 몇몇 다른 object 의 subobject 보다는) 로써 알려진다면 이 list 내의 요소가 복사될 것이다. 완료된 object 를 무엇으로 선정할지는 fixed_type_p 에 의해 수행된다.
TYPE_NAME field 는 이 type (GDB symbol table ouput 을 위한) 에 대해 프로그램에서 사용되는 이름에 대한 정보를 포함한다. typedef 들과 같은 type 들을 위한 TYPE_DECL node 이거나, tag 를 가진 것으로 알려진 struct 들 혹은 union 들, enum 들의 경우 IDENTIFIER_NODE 이며, 특별한 이름을 가지고 있지 않는 type 에 대해서는 0 을 가진다.
이름을 가질수 있는 type 의 어떤 분류이거나 named member 들을 가지는 (예를 들면, C/C++ 내에서 tagged type 들) 어떤 type 의 분류를 위해 TYPE_CONTEXT 는 주어진 type 의 scope 를 나타내는 node 를 가르키거나, 그렇지 않고 type 이 "file scope" 를 가진다면 NULL_TREE 일 것이다. 대부분의 type 들을 위해, 이것은 BLOCK node 혹은 FUNCTION_DECL node 를 가르칠 것이지만, 이것은 FUNCTION_TYPE node (scope 가 몇몇 function type specification 의 formal parameter list 로 제한되는 type 들을 위한) 일 수 있고 RECORD_TYPE node 혹은 UNION_TYPE, (C++ "member" type 들을 위한) QUAL_UNION_TYPE 를 가르킬 수 있다. Non-tagged-type 들을 위해, TYPE_CONTEXT 는 특별히 어떤 것을 설정해 줄 필요는 없는데, 이것은 어떤 type 분류 (예를 들면, array type 혹은 function type) 의 것이고 또한 자체적으로 이름을 가질수 없거나, named member 들을 가질 수 없는 type 은 se 당 "scope" 를 실제로 가지지 않기 때문이다.
TREE_CHAIN field 는 ENUMERAL_TYPE node 와 RECORD_TYPE, UNION_TYPE, QUAL_UNION_TYPE 들에 대한 이름으로 forward-reference 들 로써 사용된다; 아래를 참고하라.

VOID_TYPE t 0

C 에서의 void type

```
<void_type 0x4016a770 VOID
    align 1 symtab 0 alias set -1>
<void_type 0x4017a770 void VOID
    align 8 symtab 0 alias set -1
    pointer_to_this <pointer_type 0x4017a7e0>>
```

INTEGER_TYPE t 0

C 에서의 char 를 포함하여 모든 언어에서의 정수형 type 들. 또한 다른 분리된 type 들의 하위 범위에도 사용됩니다. Component 들인 TYPE_MIN_VALUE 와 TYPE_MAX_VALUE (expressions, inclusive) , TYPE_PRECISION (이 type 에서 사용되는 bit 들의 갯수) 를 가지고 있다. Pascal 에서 subrange type 일 경우, 이것의 TREE_TYPE 이 supertype (다른 INTEGER_TYPE 혹은 ENUMERAL_TYPE, CHAR_TYPE, BOOLEAN_TYPE) 을 가르킬 것이다. 그렇지 않으면 TREE_TYPE 은 0 이다.

```
 <integer_type 0x401669a0 unsigned DI
    size <integer_cst 0x40163600
```

```
            type <integer_type 0x401660e0> constant 64>
   unit size <integer_cst 0x40163720
            type <integer_type 0x40166070> constant 8>
   align 64 symtab 0 alias set -1 precision 64
   min <integer_cst 0x401639e0 0>
   max <integer_cst 0x40163a00 -1>>
<integer_type 0x40166a10 unsigned TI
   size <integer_cst 0x40163740
            type <integer_type 0x401660e0> constant 128>
   unit size <integer_cst 0x401638e0
                type <integer_type 0x40166070> constant 16>
   align 128 symtab 0 alias set -1 precision 128
   min <integer_cst 0x40163a20 0>
   max <integer_cst 0x40163a40 -1>>
<integer_type 0x4016a310 DI
   size <integer_cst 0x40163600
            type <integer_type 0x401660e0> constant 64>
   unit size <integer_cst 0x40163720
                type <integer_type 0x40166070> constant 8>
   align 64 symtab 0 alias set -1 precision 64
   min <integer_cst 0x40163a60 0x8000000000000000>
   max <integer_cst 0x40163a80 0x7fffffffffffffff>>
<integer_type 0x4016a3f0 unsigned DI
   size <integer_cst 0x40163600
            type <integer_type 0x401660e0> constant 64>
   unit size <integer_cst 0x40163720
                type <integer_type 0x40166070> constant 8>
   align 64 symtab 0 alias set -1 precision 64
   min <integer_cst 0x40163aa0 0>
   max <integer_cst 0x40163ac0 -1>>
<integer_type 0x4016a540 unsigned int unsigned sizetype DI
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 unsigned int> constant 64>
   unit size <integer_cst 0x40163b20
                type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 64 symtab 0 alias set -1 precision 36
   min <integer_cst 0x40163ae0 0>
   max <integer_cst 0x40163b00 0xffffffff>>
<integer_type 0x4016a5b0 SI
   size <integer_cst 0x40163b80
            type <integer_type 0x4016a540 unsigned int> constant 32>
   unit size <integer_cst 0x40163be0
                type <integer_type 0x4016a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40163ba0 -2147483648>
   max <integer_cst 0x40163bc0 2147483647>>
<integer_type 0x4016a690 DI
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 unsigned int> constant 64>
   unit size <integer_cst 0x40163b20
                type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 64 symtab 0 alias set -1 precision 36
```

```
   min <integer_cst 0x40163c20 0xffffffff800000000>
   max <integer_cst 0x40163c40 0x7fffffff>>
```

REAL_TYPE t 0

C 의 float 와 double. 다른 floating type 들은 machine mode 로 구분되거나 TYPE_SIZE 와 TYPE_PRECISION 으로 구분되어집니다.

```
<real_type 0x4016a930 SF
   size <integer_cst 0x40163b80
            type <integer_type 0x4016a540 bit_size_type> constant 32>
   unit size <integer_cst 0x40163be0
                type <integer_type 0x4016a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32>
<real_type 0x4016a9a0 DF
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 bit_size_type> constant 64>
   unit size <integer_cst 0x40163b20
                type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 64 symtab 0 alias set -1 precision 64>
<real_type 0x4016aa10 XF
   size <integer_cst 0x40163d00
            type <integer_type 0x4016a540 bit_size_type> constant 96>
   unit size <integer_cst 0x40163d40
                type <integer_type 0x4016a4d0 unsigned int> constant 12>
   align 32 symtab 0 alias set -1 precision 96>
```

COMPLEX_TYPE t 0

Complex number type 들. TREE_TYPE field 는 실수와 허수 부분의 data type 이다.

```
<complex_type 0x4016aa80
   type <integer_type 0x40166380 int SI
       size <integer_cst 0x40163540 constant 32>
       unit size <integer_cst 0x401635e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401635a0 -2147483648>
       max <integer_cst 0x401635c0 2147483647>>
   CSI
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 bit_size_type> constant 64>
   unit size <integer_cst 0x40163b20
                type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 32 symtab 0 alias set -1>
<complex_type 0x4016aaf0
   type <real_type 0x4016a930 SF
       size <integer_cst 0x40163b80 constant 32>
       unit size <integer_cst 0x40163be0 constant 4>
       align 32 symtab 0 alias set -1 precision 32>
   SC
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 bit_size_type> constant 64>
   unit size <integer_cst 0x40163b20
```

```
                type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 32 symtab 0 alias set -1>
<complex_type 0x4016ab60
   type <real_type 0x4016a9a0 DF
       size <integer_cst 0x40163900 constant 64>
       unit size <integer_cst 0x40163b20 constant 8>
       align 64 symtab 0 alias set -1 precision 64>
   DC
   size <integer_cst 0x40163de0
            type <integer_type 0x4016a540 bit_size_type> constant 128>
   unit size <integer_cst 0x40163e00
                type <integer_type 0x4016a4d0 unsigned int> constant 16>
   align 64 symtab 0 alias set -1>
<complex_type 0x4016abd0
   type <real_type 0x4016aa10 XF
       size <integer_cst 0x40163d00 constant 96>
       unit size <integer_cst 0x40163d40 constant 12>
       align 32 symtab 0 alias set -1 precision 96>
   XC
   size <integer_cst 0x40163e20
            type <integer_type 0x4016a540 bit_size_type> constant 192>
   unit size <integer_cst 0x40163e40
                type <integer_type 0x4016a4d0 unsigned int> constant 24>
   align 32 symtab 0 alias set -1>
```

VECTOR_TYPE t 0

Vector type 들. TREE_TYPE field 는 vector element 들의 data type 입니다.

```
<vector_type 0x4016ad20
   type <integer_type 0x40166930 unsigned SI
       size <integer_cst 0x40163540 constant 32>
       unit size <integer_cst 0x401635e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401639a0 0>
       max <integer_cst 0x401639c0 4294967295>>
   unsigned V4SI
   size <integer_cst 0x40163de0
       type <integer_type 0x4016a540 bit_size_type> constant 128>
   unit size <integer_cst 0x40163e00
       type <integer_type 0x4016a4d0 unsigned int> constant 16>
   align 128 symtab 0 alias set -1>
<vector_type 0x4016b000
   type <integer_type 0x40166930 unsigned SI
       size <integer_cst 0x40163540 constant 32>
       unit size <integer_cst 0x401635e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401639a0 0>
       max <integer_cst 0x401639c0 4294967295>
       pointer_to_this <pointer_type 0x4016ae00>>
   unsigned V2SI
   size <integer_cst 0x40163900
            type <integer_type 0x4016a540 bit_size_type> constant 64>
```

```
    unit size <integer_cst 0x40163b20
                  type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1>
 <vector_type 0x4016b230
    type <integer_type 0x401668c0 unsigned HI
        size <integer_cst 0x40163440 constant 16>
        unit size <integer_cst 0x40163520 constant 2>
        align 16 symtab 0 alias set -1 precision 16
        min <integer_cst 0x40163960 0>
        max <integer_cst 0x40163980 65535>>
    unsigned V4HI
    size <integer_cst 0x40163900
             type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
                  type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1>
 <vector_type 0x4016b4d0
    type <integer_type 0x40166850 unsigned QI
        size <integer_cst 0x401633e0 constant 8>
        unit size <integer_cst 0x40163400 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x40163920 0>
        max <integer_cst 0x40163940 255>>
    unsigned V8QI
    size <integer_cst 0x40163900
             type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
                  type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1>
```

ENUMERAL_TYPE t 0

C enums. Type node 는 INTEGER_TYPE node 와 거의 같다. enum type 의 값에 대한 symbol 들은 CONST_DECL node 들로 정의되지만 type 은 그것을 가르키지는 않는다; 하지만, TYPE_VALUES 는 각 element 의 TREE_PURPOSE 가 이름이고 TREE_VALUE 가 (INTEGER_CST node 인) 값인 list 이다.

enum 으로 명해진 foo 가 아직 정의되지 않았을 때, forward reference 'enum foo' 는 그것의 TYPE_SIZE 를 0 (NULL pointer) 을 갖는다. Tag name 은 TYPE_NAME field 내에 있다. 만약 type 이 나중에 정의된다면, 보통 field 들은 채워질 것이다. RECORD_TYPE 와 UNION_TYPE, QUAL_UNION_TYPE forward ref 들은 비슷하게 취급된다.

```
 <enumeral_type 0x401a29a0 processor_type unsigned type_0 SI
    size <integer_cst 0x40179540
             type <integer_type 0x4017a540 bit_size_type> constant 32>
    unit size <integer_cst 0x401795a0
                  type <integer_type 0x4017a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1 precision 32
    min <integer_cst 0x40179620 0>
    max <integer_cst 0x4019db20 7>
    chain <type_decl 0x401a2a10>>
 <enumeral_type 0x401a2e00 fpmath_unit unsigned type_0 SI
    size <integer_cst 0x40179540
```

```
            type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
                type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40179640 1>
   max <integer_cst 0x4019dbc0 2>
   chain <type_decl 0x401a2e70>>
<enumeral_type 0x401a4150 reg_class unsigned type_0 SI
   size <integer_cst 0x40179540
            type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
                type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40179620 0>
   max <integer_cst 0x401cc220 25>
   chain <type_decl 0x401a41c0>>
<enumeral_type 0x401d02a0 ix86_builtins unsigned type_0 SI
   size <integer_cst 0x40179540
            type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
                type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40179620 0>
   max <integer_cst 0x401dd040 182>
   chain <type_decl 0x401d0310>>
<enumeral_type 0x401dcb60 cmodel unsigned type_0 SI
   size <integer_cst 0x40179540
            type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
                type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40179620 0>
   max <integer_cst 0x401dd540 5>
   chain <type_decl 0x401dcbd0>>
<enumeral_type 0x401e1000 asm_dialect unsigned type_0 SI
   size <integer_cst 0x40179540
            type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
                type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1 precision 32
   min <integer_cst 0x40179620 0>
   max <integer_cst 0x40179640 1>
   chain <type_decl 0x401e1070>>
```

BOOLEAN_TYPE t 0

Pascal 의 boolean type (true 혹은 false 만 값입니다.); 특별한 field 들은 필요하지 않습니다.

```
<boolean_type 0x40188a80 _Bool unsigned QI
   size <integer_cst 0x401796a0
       type <integer_type 0x4017a540 bit_size_type> constant 8>
   unit size <integer_cst 0x40179520
       type <integer_type 0x4017a4d0 unsigned int> constant 1>
```

```
    align 8 symtab 0 alias set -1 precision 1
    min <integer_cst 0x40179ee0 0>
    max <integer_cst 0x40179f20 1>>
```

CHAR_TYPE t 0

Pascal 에서의 CHAR; C 에서는 사용되지 않습니다. 특별한 field 들은 필요하지 않습니다.

C 용 예제가 없습니다.

POINTER_TYPE t 0

모든 pointer-to-x type 들은 code POINTER_TYPE 를 가진다. TREE_TYPE 는 가르켜진 type 에 대한 node 를 가르킨다.

```
 <pointer_type 0x4016a7e0
    type <void_type 0x4016a770 VOID
        align 8 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x4016a7e0>>
    unsigned SI
    size <integer_cst 0x40163b80
             type <integer_type 0x4016a540 bit_size_type> constant 32>
    unit size <integer_cst 0x40163be0
                  type <integer_type 0x4016a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1>
 <pointer_type 0x4016ae00
    type <integer_type 0x40166930 unsigned SI
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401639a0 0>
        max <integer_cst 0x401639c0 4294967295>
        pointer_to_this <pointer_type 0x4016ae00>>
    unsigned SI
    size <integer_cst 0x40163b80
             type <integer_type 0x4016a540 bit_size_type> constant 32>
    unit size <integer_cst 0x40163be0
                  type <integer_type 0x4016a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1>
e <pointer_type 0x4016e620
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401635a0 -2147483648>
        max <integer_cst 0x401635c0 2147483647>
        pointer_to_this <pointer_type 0x4016e620>>
    unsigned SI
    size <integer_cst 0x40163b80
             type <integer_type 0x4016a540 bit_size_type> constant 32>
    unit size <integer_cst 0x40163be0
                  type <integer_type 0x4016a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1>
 <pointer_type 0x4016e770
```

```
type <integer_type 0x4016e700 char readonly QI
    size <integer_cst 0x401633e0 constant 8>
    unit size <integer_cst 0x40163400 constant 1>
    align 8 symtab 0 alias set -1 precision 8
    min <integer_cst 0x401634a0 -128>
    max <integer_cst 0x401634c0 127>
    pointer_to_this <pointer_type 0x4016e770>>
unsigned SI
size <integer_cst 0x40163b80
         type <integer_type 0x4016a540 bit_size_type> constant 32>
unit size <integer_cst 0x40163be0
              type <integer_type 0x4016a4d0 unsigned int> constant 4>
align 32 symtab 0 alias set -1>
```

OFFSET_TYPE t 0

Offset 은 object 에 상대적인 pointer 이다. TREE_TYPE field 는 offset 에서 object 의 type 이다. TYPE_OFFSET_BASETYPE 는 offset 에 상응하는 object 의 type 에 대한 node 를 가르킨다.

C 용 예제가 없습니다.

REFERENCE_TYPE t 0

Reference 는 자동으로 값이 어떤 것을 가르키게 강요하는 것을 제외하고는 pointer 와 같다. C++ 에서 사용됨.

```
<reference_type 0x4016ea80
   type <pointer_type 0x4016acb0 __builtin_va_list
       type <integer_type 0x40166230 char QI
           size <integer_cst 0x401633e0 constant 8>
           unit size <integer_cst 0x40163400 constant 1>
           align 8 symtab 0 alias set -1 precision 8
           min <integer_cst 0x401634a0 -128>
           max <integer_cst 0x401634c0 127>
           pointer_to_this <pointer_type 0x4016ac40>>
       unsigned SI
       size <integer_cst 0x40163b80 constant 32>
       unit size <integer_cst 0x40163be0 constant 4>
       align 32 symtab 0 alias set -1
       reference_to_this <reference_type 0x4016ea80>>
   unsigned SI
   size <integer_cst 0x40163b80 32>
   unit size <integer_cst 0x40163be0 4>
   align 32 symtab 0 alias set -1>

<reference_type 0x4017da80
   type <pointer_type 0x4017acb0 __builtin_va_list
       type <integer_type 0x40166230 char QI
           size <integer_cst 0x40176da0 constant 8>
           unit size <integer_cst 0x40176dc0 constant 1>
           align 8 symtab 0 alias set -1 precision 8
           min <integer_cst 0x40176e60 -128>
           max <integer_cst 0x40176e80 127>
```

```
            pointer_to_this <pointer_type 0x4017ac40>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        reference_to_this <reference_type 0x4017da80>>
    unsigned SI
    size <integer_cst 0x40179540 32>
    unit size <integer_cst 0x401795a0 4>
    align 32 symtab 0 alias set -1>

<reference_type 0x4017da80
    type <pointer_type 0x4017acb0 __builtin_va_list
        type <integer_type 0x40166230 char QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017ac40>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        reference_to_this <reference_type 0x4017da80>>
    unsigned SI
    size <integer_cst 0x40179540 32>
    unit size <integer_cst 0x401795a0 4>
    align 32 symtab 0 alias set -1>

<reference_type 0x4017da80
    type <pointer_type 0x4017acb0 __builtin_va_list
        type <integer_type 0x40166230 char QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017ac40>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        reference_to_this <reference_type 0x4017da80>>
    unsigned SI
    size <integer_cst 0x40179540 32>
    unit size <integer_cst 0x401795a0 4>
    align 32 symtab 0 alias set -1>

<reference_type 0x4017da80
    type <pointer_type 0x4017acb0 __builtin_va_list
        type <integer_type 0x40166230 char QI
```

```
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017ac40>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        reference_to_this <reference_type 0x4017da80>>
    unsigned SI
    size <integer_cst 0x40179540 32>
    unit size <integer_cst 0x401795a0 4>
    align 32 symtab 0 alias set -1>
```

METHOD_TYPE t 0

METHOD_TYPE 는 여분의 첫 argument 로 "자신" 을 가지는 함수의 type 이다. 여기서 argument 는 선언된 argument list 에는 존재하지 않는 것이다. TREE_TYPE 은 이 method 의 return type 이다. TYPE_METHOD_BASETYPE 는 "자신" 의 type 이다. TYPES 는 숨겨인 argument 인 "자신" 을 포함하는 실제 argument list 이다.

C 용 예제가 없습니다.

FILE_TYPE t 0

Pascal 용으로 사용됨; 아직 확실히 사용이 정의되지 않았다.

C 용 예제가 없습니다.

ARRAY_TYPE t 0

배열들의 type 들. 특별한 field 들:

```
TREE_TYPE          배열 element 의 type.
TYPE_DOMAIN        어떤 순으로 나열된 type.
                     그것의 값들의 범위는 배열의 길이를 나타낸다.
TYPE_SEP                 하나의 elt 부터 다음까지의 unit 들을 위한 expression.
TYPE_SEP_UNIT      이전 unit 의 bit 들의 갯수.
```

Field TYPE_POINTER_TO (TREE_TYPE (array_type)) 는 항상 0 이 아니며 C 에서는 해당 array type 의 값으로 강제로 가르키게 한 type 을 잡고 있다. TYPE_STRING_FLAG 는 언어들 상에서 (Chill 과 같은) 구분을 만드는 문자열 (char 들의 배열과 대조적인) 을 가르킨다.

```
 <array_type 0x4016ae70
    type <integer_type 0x40166930 unsigned SI
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401639a0 0>
        max <integer_cst 0x401639c0 4294967295>
        pointer_to_this <pointer_type 0x4016ae00>>
    BLK
    size <integer_cst 0x40163de0
```

```
        type <integer_type 0x4016a540 bit_size_type> constant 128>
    unit size <integer_cst 0x40163e00
        type <integer_type 0x4016a4d0 unsigned int> constant 16>
    align 32 symtab 0 alias set -1
    domain <integer_type 0x4016ad90
        type <integer_type 0x4016a4d0 unsigned int unsigned sizetype
            SI
            size <integer_cst 0x40163540 32>
            unit size <integer_cst 0x401635e0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40163620 0>
            max <integer_cst 0x40163640 4294967295>>
        SI
        size <integer_cst 0x40163540 32>
        unit size <integer_cst 0x401635e0 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40163c00 0>
        max <integer_cst 0x40163d80 3>>>
<array_type 0x4016b0e0
    type <integer_type 0x40166930 unsigned SI
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401639a0 0>
        max <integer_cst 0x401639c0 4294967295>
        pointer_to_this <pointer_type 0x4016ae00>>
    DI
    size <integer_cst 0x40163900
        type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 32 symtab 0 alias set -1
    domain <integer_type 0x4016b070
        type <integer_type 0x4016a4d0 unsigned int unsigned sizetype
            SI
            size <integer_cst 0x40163540 32>
            unit size <integer_cst 0x401635e0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40163620 0>
            max <integer_cst 0x40163640 4294967295>>
        SI size <integer_cst 0x40163540 32> unit size <integer_cst 0x401635e0 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40163c00 0>
        max <integer_cst 0x40163b60 1>>>
<array_type 0x4016b380
    type <integer_type 0x401668c0 unsigned HI
        size <integer_cst 0x40163440 constant 16>
        unit size <integer_cst 0x40163520 constant 2>
        align 16 symtab 0 alias set -1 precision 16
        min <integer_cst 0x40163960 0>
        max <integer_cst 0x40163980 65535>
        pointer_to_this <pointer_type 0x4016b310>>
```

```
    DI
    size <integer_cst 0x40163900
        type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 16 symtab 0 alias set -1
    domain <integer_type 0x4016ad90
        type <integer_type 0x4016a4d0 unsigned int unsigned sizetype SI
            size <integer_cst 0x40163540 constant 32>
            unit size <integer_cst 0x401635e0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40163620 0>
            max <integer_cst 0x40163640 4294967295>>
        SI size <integer_cst 0x40163540 32> unit size <integer_cst 0x401635e0 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40163c00 0>
        max <integer_cst 0x40163d80 3>>>
<array_type 0x4016e850
    type <integer_type 0x40166460 long int SI
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40163660 -2147483648>
        max <integer_cst 0x40163680 2147483647>
        pointer_to_this <pointer_type 0x4016e7e0>>
    BLK
    size <integer_cst 0x4016c3e0
        type <integer_type 0x4016a540 bit_size_type> constant 6432>
    unit size <integer_cst 0x4016c420
        type <integer_type 0x4016a4d0 unsigned int> constant 804>
    align 32 symtab 0 alias set -1
    domain <integer_type 0x4016e540
        type <integer_type 0x4016a4d0 unsigned int unsigned sizetype
            SI
            size <integer_cst 0x40163540 32>
            unit size <integer_cst 0x401635e0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40163620 0>
            max <integer_cst 0x40163640 4294967295>>
        SI size <integer_cst 0x40163540 32> unit size <integer_cst 0x401635e0 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40163c00 0>
        max <integer_cst 0x4016c0e0 200>>>
<array_type 0x4017b620
    type <integer_type 0x40166850 unsigned QI
        size <integer_cst 0x40176da0 constant 8>
        unit size <integer_cst 0x40176dc0 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401792e0 0>
        max <integer_cst 0x40179300 255>
        pointer_to_this <pointer_type 0x4017b5b0>>
    DI
```

```
size <integer_cst 0x401792c0
    type <integer_type 0x4017a540 bit_size_type> constant 64>
unit size <integer_cst 0x401794e0
    type <integer_type 0x4017a4d0 unsigned int> constant 8>
align 8 symtab 0 alias set -1
domain <integer_type 0x4017b540
    type <integer_type 0x4017a4d0 unsigned int unsigned sizetype SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40176fe0 0>
        max <integer_cst 0x40179000 4294967295>>
    SI size <integer_cst 0x40176f00 32> unit size <integer_cst 0x40176fa0 4>
    align 32 symtab 0 alias set -1 precision 32
    min <integer_cst 0x401795c0 0>
    max <integer_cst 0x401798e0 7>>>
```

SET_TYPE t 0

Pascal 용 set (집합) 들의 type.
특별한 field 들은 array type 의 것들과 같다. Target type 은 항상 boolean type 이다. Chill 에서는 bitstring 들과 powerset 들 둘다 사용된다; TYPE_STRING_FLAG 는 bitstring 을 지정한다.

C 용 예제가 없습니다.

RECORD_TYPE t 0

C 에서는 struct, 혹은 Pascal 에서는 record.

특별한 field 들:

TYPE_FIELDS	struct 와 VAR_DECL 들의 field 에 대한 FIELD_DECL 들의 chain. record-scope 변수, type 들, 열거자(계수자) 들에 대한 TYPE_DECL 들과 CONST_DECL 들.

Pascal 에 대해서는 추가할 필요가 거의 없을 것이다.

어떻게 struct tag forward references 가 C 에서 다루어 지는지에 대해서는 ENUMERAL_TYPE 앞의 위 comment 를 봐라.

```
<record_type 0x4016aee0 BLK
   size <integer_cst 0x40163de0
       type <integer_type 0x4016a540 bit_size_type> constant 128>
   unit size <integer_cst 0x40163e00
       type <integer_type 0x4016a4d0 unsigned int> constant 16>
   align 32 symtab 0 alias set -1
   fields <field_decl 0x4016af50 f
       type <array_type 0x4016ae70 type <integer_type 0x40166930>
           BLK
           size <integer_cst 0x40163de0 128>
           unit size <integer_cst 0x40163e00 16>
           align 32 symtab 0 alias set -1
           domain <integer_type 0x4016ad90>>
       BLK file <built-in> line 0
```

```
        size <integer_cst 0x40163de0 128>
        unit size <integer_cst 0x40163e00 16>
        align 32 offset_align 128
        offset <integer_cst 0x40163c00 constant 0>
        bit offset <integer_cst 0x40163cc0 constant 0>
        context <record_type 0x4016aee0>
        arguments <integer_cst 0x40163c00 0>>>
<record_type 0x4016b150 DI
   size <integer_cst 0x40163900
       type <integer_type 0x4016a540 bit_size_type> constant 64>
   unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 32 symtab 0 alias set -1
   fields <field_decl 0x4016b1c0 f
       type <array_type 0x4016b0e0 type <integer_type 0x40166930>
           DI
           size <integer_cst 0x40163900 64>
           unit size <integer_cst 0x40163b20 8>
           align 32 symtab 0 alias set -1
           domain <integer_type 0x4016b070>>
       DI file <built-in> line 0
       size <integer_cst 0x40163900 64>
       unit size <integer_cst 0x40163b20 8>
       align 32 offset_align 128
       offset <integer_cst 0x40163c00 constant 0>
       bit offset <integer_cst 0x40163cc0 constant 0>
       context <record_type 0x4016b150>
       arguments <integer_cst 0x40163c00 0>>>
<record_type 0x4016b3f0 DI
   size <integer_cst 0x40163900
       type <integer_type 0x4016a540 bit_size_type> constant 64>
   unit size <integer_cst 0x40163b20
       type <integer_type 0x4016a4d0 unsigned int> constant 8>
   align 16 symtab 0 alias set -1
   fields <field_decl 0x4016b460 f
       type <array_type 0x4016b380 type <integer_type 0x401668c0>
           DI
           size <integer_cst 0x40163900 64>
           unit size <integer_cst 0x40163b20 8>
           align 16 symtab 0 alias set -1
           domain <integer_type 0x4016ad90>>
       DI file <built-in> line 0
       size <integer_cst 0x40163900 64>
       unit size <integer_cst 0x40163b20 8>
       align 16 offset_align 128
       offset <integer_cst 0x40163c00 constant 0>
       bit offset <integer_cst 0x40163cc0 constant 0>
       context <record_type 0x4016b3f0>
       arguments <integer_cst 0x40163c00 0>>>
<record_type 0x40191bd0 processor_costs tree_1 type_0 BLK
   size <integer_cst 0x4019d880
       type <integer_type 0x4017a540 bit_size_type> constant 1280>
```

```
    unit size <integer_cst 0x4019d860
        type <integer_type 0x4017a4d0 unsigned int> constant 160>
    align 32 symtab 0 alias set -1
    fields <field_decl 0x40191cb0 add
        type <integer_type 0x40166380 int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        readonly SI file config/i386/i386.h line 54
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
        align 32 offset_align 128
        offset <integer_cst 0x401795c0 constant 0>
        bit offset <integer_cst 0x40179680 constant 0>
        context <record_type 0x40191bd0 processor_costs>
        arguments <integer_cst 0x401795c0 0>
        chain <field_decl 0x40191d20 lea type <integer_type 0x40166380 int>
            readonly SI file config/i386/i386.h line 55
            size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 offset_align 128
            offset <integer_cst 0x401795c0 0>
            bit offset <integer_cst 0x40176f00 32>
            context <record_type 0x40191bd0 processor_costs>
            arguments <integer_cst 0x401795c0 0>
            chain <field_decl 0x40191d90 shift_var>>>
    chain <type_decl 0x40191c40>>
<record_type 0x401a4d90 ix86_args type_0 BLK
    size <integer_cst 0x4019d8e0
        type <integer_type 0x4017a540 bit_size_type> constant 224>
    unit size <integer_cst 0x401cc280
        type <integer_type 0x4017a4d0 unsigned int> constant 28>
    align 32 symtab 0 alias set -1
    fields <field_decl 0x401a4e70 words
        type <integer_type 0x40166380 int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        SI file config/i386/i386.h line 1676
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
        align 32 offset_align 128
        offset <integer_cst 0x401795c0 constant 0>
        bit offset <integer_cst 0x40179680 constant 0>
        context <record_type 0x401a4d90 ix86_args>
        arguments <integer_cst 0x401795c0 0>
```

```
        chain <field_decl 0x401a4ee0 nregs type <integer_type 0x40166380 int>
            SI file config/i386/i386.h line 1677
            size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 offset_align 128
            offset <integer_cst 0x401795c0 0>
            bit offset <integer_cst 0x40176f00 32>
            context <record_type 0x401a4d90 ix86_args>
            arguments <integer_cst 0x401795c0 0>
            chain <field_decl 0x401a4f50 regno>>>
    chain <type_decl 0x401a4e00>>
```

UNION_TYPE t 0

C 에서 사용되지 않는 TREE node 입니다.

C 에서의 union. Field 들의 offset 들이 전부 0 일것을 제외하곤 struct 와 비슷.

어떻게 struct tag forward references 가 C 에서 다루어 지는지에 대해서는 ENUMERAL_TYPE 앞의 위 comment 를 봐라.

```
 <union_type 0x402033f0 type_0 SI
    size <integer_cst 0x40179540
        type <integer_type 0x4017a540 bit_size_type> constant 32>
    unit size <integer_cst 0x401795a0
        type <integer_type 0x4017a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1
    fields <field_decl 0x40203230 __wch
        type <integer_type 0x40203150 wint_t unsigned SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176fe0 0>
            max <integer_cst 0x40179000 4294967295>>
        unsigned in_system_header SI file /usr/include/wchar.h line 72
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
        align 32 offset_align 128
        offset <integer_cst 0x401795c0 constant 0>
        bit offset <integer_cst 0x40179680 constant 0>
        context <union_type 0x402033f0>
        arguments <integer_cst 0x401795c0 0>
        chain <field_decl 0x40203380 __wchb type <array_type 0x40203310>
            in_system_header SI file /usr/include/wchar.h line 73
            size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
            align 8 offset_align 128 offset <integer_cst 0x401795c0 0>
            bit offset <integer_cst 0x40179680 0>
            context <union_type 0x402033f0> arguments <integer_cst 0x401795c0 0>>>
    chain <type_decl 0x40203460>>
 <union_type 0x4021ab60 type_0 BLK
    size <integer_cst 0x4019d960
        type <integer_type 0x4017a540 bit_size_type> constant 352>
    unit size <integer_cst 0x401ddf40
        type <integer_type 0x4017a4d0 unsigned int> constant 44>
```

```
    align 32 symtab 0 alias set -1
    fields <field_decl 0x4021a8c0 __cd
        type <record_type 0x4021a460 __gconv_info type_0 BLK
            size <integer_cst 0x401792c0 constant 64>
            unit size <integer_cst 0x401794e0 constant 8>
            align 32 symtab 0 alias set -1
            fields <field_decl 0x4021a540 __nsteps>
            pointer_to_this <pointer_type 0x4021a770>
            chain <type_decl 0x4021a4d0>>
        in_system_header BLK file /usr/include/_G_config.h line 47
        size <integer_cst 0x401792c0 64>
        unit size <integer_cst 0x401794e0 8>
        align 32 offset_align 128
        offset <integer_cst 0x401795c0 constant 0>
        bit offset <integer_cst 0x40179680 constant 0>
        context <union_type 0x4021ab60>
        arguments <integer_cst 0x401795c0 0>
        chain <field_decl 0x4021aaf0 __combined type <record_type 0x4021aa10>
            in_system_header BLK file /usr/include/_G_config.h line 52
            size <integer_cst 0x4019d960 352> unit size <integer_cst 0x401ddf40 44>
            align 32 offset_align 128 offset <integer_cst 0x401795c00>
            bit offset <integer_cst 0x40179680 0>
            context <union_type 0x4021ab60>
            arguments <integer_cst 0x401795c0 0>>>
    chain <type_decl 0x4021abd0>>
<union_type 0x4028bee0 wait type_0 SI
    size <integer_cst 0x40179540
        type <integer_type 0x4017a540 bit_size_type> constant 32>
    unit size <integer_cst 0x401795a0
        type <integer_type 0x4017a4d0 unsigned int> constant 4>
    align 32 symtab 0 alias set -1
    fields <field_decl 0x40291000 w_status
        type <integer_type 0x40166380 int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        in_system_header SI file /usr/include/bits/waitstatus.h line 67
        size <integer_cst 0x40176f00 32> unit size <integer_cst 0x40176fa0 4>
        align 32 offset_align 128
        offset <integer_cst 0x401795c0 constant 0>
        bit offset <integer_cst 0x40179680 constant 0>
        context <union_type 0x4028bee0 wait>
        arguments <integer_cst 0x401795c0 0>
        chain <field_decl 0x40291310 __wait_terminated
            type <record_type 0x40291230>
            in_system_header SI file /usr/include/bits/waitstatus.h line 82
            size <integer_cst 0x40179540 32>
            unit size <integer_cst 0x401795a0 4>
            align 32 offset_align 128 offset <integer_cst 0x401795c0 0>
```

```
            bit offset <integer_cst 0x40179680 0>
            context <union_type 0x4028bee0 wait>
            arguments <integer_cst 0x401795c0 0>
            chain <field_decl 0x402915b0 __wait_stopped>>>
   chain <type_decl 0x4028bf50>>
   align 32 symtab 0 alias set -1>
<union_type 0x40291770 type_0 SI
   size <integer_cst 0x40179540
       type <integer_type 0x4017a540 bit_size_type> constant 32>
   unit size <integer_cst 0x401795a0
       type <integer_type 0x4017a4d0 unsigned int> constant 4>
   align 32 symtab 0 alias set -1
   fields <field_decl 0x40291690 __uptr
       type <pointer_type 0x40291620 type <union_type 0x4028bee0 wait>
           unsigned SI
           size <integer_cst 0x40179540 32>
           unit size <integer_cst 0x401795a0 4>
           align 32 symtab 0 alias set -1>
       unsigned in_system_header SI file /usr/include/stdlib.h line 70
       size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
       align 32 offset_align 128
       offset <integer_cst 0x401795c0 constant 0>
       bit offset <integer_cst 0x40179680 constant 0>
       context <union_type 0x40291770>
       arguments <integer_cst 0x401795c0 0>
       chain <field_decl 0x40291700 __iptr type <pointer_type 0x4017d620>
           unsigned in_system_header SI file /usr/include/stdlib.h line 71
           size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
           align 32 offset_align 128 offset <integer_cst 0x401795c0 0>
            bit offset <integer_cst 0x40179680 0>
            context <union_type 0x40291770> arguments <integer_cst 0x401795c0 0>>>
   chain <type_decl 0x402917e0>>
```

QUAL_UNION_TYPE t 0

각 FIELD_DECL 내의 DECL_QUALIFIER 의 표현식이 무슨 union 을 포함하고 있는지 결정하는 것을 제외하고는 UNION_TYPE 와 비슷하다. DECL_QUALIFIER expression 이 true 인 처음 field 는 union 을 차지하고 있는 것으로 생각 되어진다.

C 용 예제가 없습니다.

FUNCTION_TYPE t 0

함수들의 type. 특별한 field 들:

```
TREE_TYPE                   반환되는 값의 type.
TYPE_ARG_TYPES              예상된 argument 들의 type 들의 list.
                            이 list 는 TREE_LIST node 들로 구성된다.
```

언어들내에서는 "Procedures" 의 type 들, 그들은 code FUNCTION_TYPE 를 또한 가지고 있지만 TREE_TYPE 이 0 혹은 void type 인 함수와는 다르다.

```
<function_type 0x4016e8c0
   type <integer_type 0x40166380 int SI
```

```
        size <integer_cst 0x40163540 constant 32>
        unit size <integer_cst 0x401635e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401635a0 -2147483648>
        max <integer_cst 0x401635c0 2147483647>
        pointer_to_this <pointer_type 0x4016e620>>
    DI
    size <integer_cst 0x40163900
        type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1>
<function_type 0x4016eb60
    type <pointer_type 0x4016a7e0
        type <void_type 0x4016a770 void VOID
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x4016a7e0>>
        unsigned SI
        size <integer_cst 0x40163b80 constant 32>
        unit size <integer_cst 0x40163be0 constant 4>
        align 32 symtab 0 alias set -1>
    DI
    size <integer_cst 0x40163900
        type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1
    arg-types <tree_list 0x40164974 value <void_type 0x4016a770 void>>>
<function_type 0x4016ed90
    type <real_type 0x4016a930 float SF
        size <integer_cst 0x40163b80 constant 32>
        unit size <integer_cst 0x40163be0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        pointer_to_this <pointer_type 0x4016bcb0>>
    DI
    size <integer_cst 0x40163900
        type <integer_type 0x4016a540 bit_size_type> constant 64>
    unit size <integer_cst 0x40163b20
        type <integer_type 0x4016a4d0 unsigned int> constant 8>
    align 64 symtab 0 alias set -1
    arg-types <tree_list 0x40164a00 value <real_type 0x4016a930 float>
        chain <tree_list 0x40164974 value <void_type 0x4016a770 void>>>>
<function_type 0x40244ee0
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40176f60 -2147483648>
        max <integer_cst 0x40176f80 2147483647>
        pointer_to_this <pointer_type 0x4017d620>>
    DI
    size <integer_cst 0x401792c0
```

```
            type <integer_type 0x4017a540 bit_size_type> constant 64>
        unit size <integer_cst 0x401794e0
            type <integer_type 0x4017a4d0 unsigned int> constant 8>
        align 64 symtab 0 alias set -1
        arg-types <tree_list 0x40245564
            value <pointer_type 0x40244b60 type <record_type 0x40244a10 obstack>
                unsigned SI
                size <integer_cst 0x40179540 constant 32>
                unit size <integer_cst 0x401795a0 constant 4>
                align 32 symtab 0 alias set -1>
            chain <tree_list 0x40245578 value <pointer_type 0x40227690>
                chain <tree_list 0x4024558c
                    value <pointer_type 0x401e92a0 __gnuc_va_list>
                    chain <tree_list 0x402455a0 value <void_type 0x4017a770 void>>>>>>
    <function_type 0x402464d0
        type <integer_type 0x40166380 int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        DI
        size <integer_cst 0x401792c0
            type <integer_type 0x4017a540 bit_size_type> constant 64>
        unit size <integer_cst 0x401794e0
            type <integer_type 0x4017a4d0 unsigned int> constant 8>
        align 64 symtab 0 alias set -1
        arg-types <tree_list 0x40245aa0
            value <pointer_type 0x40227690 type <integer_type 0x4017d700 char>
                unsigned SI
                size <integer_cst 0x40179540 constant 32>
                unit size <integer_cst 0x401795a0 constant 4>
                align 32 symtab 0 alias set -1>
            chain <tree_list 0x40245ab4 value <pointer_type 0x401e92a0 __gnuc_va_list>
                chain <tree_list 0x40245ac8 value <void_type 0x4017a770 void>>>>>
```

LANG_TYPE t 0

이것은 type 의 language-specific 종류이다.
이것의 의미는 language front end 에서 정의된다.
layout_type 은 이것을 어떻게 layout 할지 모르므로, 반드시 front-end 가 수동으로 해줘야한다.

C 용 예제가 없습니다.

표현식들

INTEGER_CST c 2

내용들은 TREE_INT_CST_LOW 와 TREE_INT_CST_HIGH field 내에 있으며, 각각은 32 bit 크기이며, 64 bit 상수 용량을 제공한다. 참고: Pascal 에서 type char 의 상수들은 INTEGER_CST 이고, Pascal 내의 nil 혹은 C 에서의 NULL 과 같은 것들은 포인터 상수들이다. C 에서의 '(int *) 1' 또한 INTEGET_CST 와 같이 결과로 나온다.

```
<integer_cst 0x401792c0 constant 64>
<integer_cst 0x401794e0 constant 8>
<integer_cst 0x40176f60 -2147483648>
<integer_cst 0x40176f80 2147483647>
<integer_cst 0x401dd340 type <integer_type 0x40166380 int> constant 53>
```

REAL_CST c 3

내용들은 TREE_REAL_CST field 에 있으며 또한 TREE_CST_RTL 에도 있다.

```
<real_cst 0x40181a00
   type <real_type 0x4016a9a0 double DF
       size <integer_cst 0x40163900 constant 64>
       unit size <integer_cst 0x40163b20 constant 8>
       align 64 symtab 0 alias set -1 precision 64>
   constant 1.4439999999999995026e0>
```

COMPLEX_CST c 3

내용들은 다른 상수 node 인 TREE_REALPART 와 TREE_IMAGPART field 들내에 있다. 또한 TREE_CST_RTL 에도 있다.

```
<complex_cst 0x4016c5e0
   type <complex_type 0x401825b0
       type <real_type 0x4016a9a0 double DF
           size <integer_cst 0x40163900 constant 64>
           unit size <integer_cst 0x40163b20 constant 8>
           align 64 symtab 0 alias set -1 precision 64>
       DC
       size <integer_cst 0x40163de0 constant 128>
       unit size <integer_cst 0x40163e00 constant 16>
       align 64 symtab 0 alias set -1>
   constant
   real <real_cst 0x40181a40 type <real_type 0x4016a9a0 double>
       constant 0.00000000000000000000e0>
   imag <real_cst 0x40181a00 type <real_type 0x4016a9a0 double>
       constant 1.3999999999999991118e0>>
<complex_cst 0x4016c600
   type <complex_type 0x401825b0
       type <integer_type 0x40166380 int SI
           size <integer_cst 0x40163540 constant 32>
           unit size <integer_cst 0x401635e0 constant 4>
           align 32 symtab 0 alias set -1 precision 32
           min <integer_cst 0x401635a0 -2147483648>
           max <integer_cst 0x401635c0 2147483647>
           pointer_to_this <pointer_type 0x4016e620>>
       CSI
       size <integer_cst 0x40163900 constant 64>
       unit size <integer_cst 0x40163b20 constant 8>
       align 32 symtab 0 alias set -1>
   constant
   real <integer_cst 0x40163c60 type <integer_type 0x40166380 int> constant 0>
   imag <integer_cst 0x4016c5e0 type <integer_type 0x40166380 int> constant 4>>
```

VECTOR_CST c 3

내용들은 TREE_VECTOR_CST_ELTS field 내에 있다.

C 용 예제가 없습니다. 적당한 예제를 찾지 못하였습니다.

STRING_CST c 3

내용들은 TREE_STRING_LENGTH 와 TREE_STRING_POINTER field 에 있으며 또한 TREE_CST_RTL 에도 있다.

```
 <string_cst 0x4030a560
    type <array_type 0x4030f2a0
        type <integer_type 0x4017d700 char readonly QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017d770>>
        BLK
        size <integer_cst 0x401dde60 constant 56>
        unit size <integer_cst 0x401798e0 constant 7>
        align 8 symtab 0 alias set -1
        domain <integer_type 0x402ae930
            type <integer_type 0x4017a4d0
                unsigned int unsigned sizetype SI
                size <integer_cst 0x40176f00 constant 32>
                unit size <integer_cst 0x40176fa0 constant 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x40176fe0 0>
                max <integer_cst 0x40179000 4294967295>>
            SI size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401795c0 0>
            max <integer_cst 0x40179840 6>>>
    readonly constant static ".ctors">
 <string_cst 0x40326280
    type <array_type 0x4017d5b0
        type <integer_type 0x40166230 char QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set 5 precision 8
            min <integer_cst 0x40176e60 -128>
            max <integer_cst 0x40176e80 127>
            pointer_to_this <pointer_type 0x4017ac40>>
        BLK
        size <integer_cst 0x40179d80 constant 1608>
        unit size <integer_cst 0x40179d20 constant 201>
        align 8 symtab 0 alias set -1
        domain <integer_type 0x4017d540
            type <integer_type 0x4017a4d0 unsigned int unsigned sizetype SI
                size <integer_cst 0x40176f00 constant 32>
```

```
                unit size <integer_cst 0x40176fa0 constant 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x40176fe0 0>
                max <integer_cst 0x40179000 4294967295>>
            SI size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401795c0 0>
            max <integer_cst 0x40179aa0 200>>>
    constant "frame_dummy">
```

선언들. 이름들을 위한 모든 참조(reference)들은 ...._DECL node 들로써 나타내어집니다. 하나의 binding context 내의 decl 들은 TREE_CHAIN 필드를 통해서 연결되어 있습니다. 각 DECL 은 IDENTIFIER_NODE 를 포함하고 있는 DECL_NAME 필드를 가지고 있습니다.
(몇몇 decl 들[대부분 label 관련]은 DECL_NAME 으로 NULL 을 가지고 있을 수 있습니다.)
DECL_CONTEXT 는 이 선언이 그것의 scope 를 가지고 있음을 나타내는 context 를 나타내는 node 를 가르킨다. FIELD_DECL 들을 위해, 이것은 field 가 어떤 것의 member 인 RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE node 이다. VAR_DECL 와 PARM_DECL, FUNCTION_DECL, LABEL_DECL, CONST_DECL node 들을 위해, 이것은 함수를 포함하는 FUNCTION_DECL 를 가르키거나, type 을 포함하는 RECORD_TYPE 혹은 UNION_TYPE 를 가르키거나, 주어진 decl 이 "file scope" 를 가지고 있다면 NULL_TREE 를 가르킨다.
DECL_ABSTRACT_ORIGIN 가 NULL 이 아닐 경우, 이 decl 이 (inlined 혹은 임시적인 확장을 위한) instance 인 것의 원래 (abstract) ...._DECL 를 가르킨다.
TREE_TYPE field 는 의미 있을 경우, object 의 data type 을 가지고 있다. LABEL_DECL 들은 data type 을 가지고 있지 않다. TYPE_DECL 에 대해, TREE_TYPE field 내용들은 이름이 이미 선언된 type 이다.
DECL_ALIGN 와 DECL_SIZE, DECL_MODE field 들은 type node 들과 마찬가지로 단지 decl node 들에서만 존재한다. 그들은 LABEL_DECL 와_DECL, CONST_DECL node 에서는 사용되지 않는다.
DECL_OFFSET 는 위치를 위한 bits offset 의 정수수를 가지고 있다.
DECL_VOFFSET 는 변수 offset 을 위한 표현식을 가지고 있다; 이것은
DECL_VOFFSET_UNIT (정수) 에 의해 곱해진다.
이것의 field 들은 FIELD_DECL 와 PARM_DECL 들내에서만 의미있다.
DECL_INITIAL 는 변수를 초기화하기 위한 값을 가지고 있거나 상수의 값을 가지고 있다. 함수에 대해서, 이것은 body (함수의 binding contour 를 나타내는 type BLOCK 의 node 이고 body 는 함수의 statement 들을 포함 한다.) 를 가지고 있다. C 에서의 LABEL_DECL 에 대해서는, 이것은 flag 이고, 만약 label 의 정의가 나타났었다면 0 이 아닌 값을 가진다.
PARM_DECL 들은 특별한 field 를 사용한다:
DECL_ARG_TYPE 는 argument 가 실제로 전달하는 type 이며, 이것은 함수 내에서의 그것의 type 과 약간 다를 수 있다.
FUNCTION_DECL 들은 네가지 특별한 field 들을 사용한다:
DECL_ARGUMENTS 는 argument 들을 위한 PARM_DECL node 들의 chain 을 가지고 있다.
DECL_RESULT 는 함수의 값을 위한 RESULT_DECL node 를 가지고 있거나, 혹은 반환할 값이 없는 함수의 경우 0 을 가지고 있는다. (Void 를 반환하는 C 함수들은 여기서 0 을 가진다.)
TREE_TYPE field 는 결과가 실제로 반환되었는지를 나타내는 type 이다. 이것은 보통 FUNCTION_DECL 의 return type 과 같지만, promotion 으로 인해 wider integer type 일 수도 있다.
DECL_FUNCTION_CODE 는 built-in 함수들에 대해서는 0 이 아닌 code 번호이다. 그것의 값은 그것이 어떤 built-in 함수를 말해주는 enum built_in_function 이다.
DECL_SOURCE_FILE 는 파일이름 문자열을 가지고 있고 DECL_SOURCE_LINE 는 줄번호를 가지고 있다. 몇몇 경우에 정의가 보이지 않는 다면 이것들은 reference 의 위치일 수 있다.
DECL_ABSTRACT 는 만약 decl 이 decl 대신에 abstract 를 나타낸다면 0 이 아닌 값을 갖는다. (예를 들면, Inline function 대신에 abstract 내에 nest 된 어떤 것).

FUNCTION_DECL d 0

아직 정확한 설명은 없습니다.

```
<function_decl 0x40293540 atoi
   type <function_type 0x4017e460
       type <integer_type 0x40166380 int SI
           size <integer_cst 0x40176f00 constant 32>
           unit size <integer_cst 0x40176fa0 constant 4>
           align 32 symtab 0 alias set 4 precision 32
```

```
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 64 symtab 0 alias set -1
        arg-types <tree_list 0x40164b2c value <pointer_type 0x4017d770>
            chain <tree_list 0x40164974 value <void_type 0x4017a770 void>>>>
    nothrow public
    in_system_header external inline defer-output
        QI file /usr/include/stdlib.h line 361
    attributes <tree_list 0x40292f3c>
    arguments <parm_decl 0x402a79a0 __nptr
        type <pointer_type 0x4017d770 type <integer_type 0x4017d700 char>
            unsigned SI
            size <integer_cst 0x40179540 constant 32>
            unit size <integer_cst 0x401795a0 constant 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x40215e00>>
        unsigned used in_system_header SI file /usr/include/stdlib.h line 360
        size <integer_cst 0x40179540 32>
        unit size <integer_cst 0x401795a0 4>
        align 32 context <function_decl 0x40293540 atoi>
        result <pointer_type 0x4017d770>
        initial <pointer_type 0x4017d770>
        (reg/v/f:SI 59) arg-type <pointer_type 0x4017d770>
        arg-type-as-written <pointer_type 0x4017d770>
        incoming-rtl (mem/f:SI
            (reg/f:SI 53 virtual-incoming-args) [7 __nptr+0 S4 A32])>
    result <result_decl 0x402a7af0 type <integer_type 0x40166380 int>
        in_system_header regdecl SI file /usr/include/stdlib.h line 361
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
        align 32 context <function_decl 0x40293540 atoi>
        (reg:SI 58)> initial <block 0x402906c0>
    (mem:QI (symbol_ref:SI ("atoi")) [0 S1 A8])
    saved-insns 0x4032fa00 chain <function_decl 0x402933f0 atof>>
<function_decl 0x40236a10 getchar
    type <function_type 0x4022c9a0
        type <integer_type 0x40166380 int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set 18 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4017d620>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 64 symtab 0 alias set -1
        arg-types <tree_list 0x4022d1b8 value <void_type 0x4017a770 void>>>
```

```
    nothrow public
    in_system_header external inline defer-output
        QI file /usr/include/bits/stdio.h line 42
    result <result_decl 0x40246850 type <integer_type 0x40166380 int>
        in_system_header regdecl SI file /usr/include/bits/stdio.h line 42
        size <integer_cst 0x40176f00 32> unit size <integer_cst 0x40176fa0 4>
        align 32 context <function_decl 0x40236a10 getchar>
        (reg:SI 58)> initial <block 0x40240900>
    (mem:QI (symbol_ref:SI ("getchar")) [0 S1 A8])
    saved-insns 0x402a9c00 chain <function_decl 0x402368c0 getc>>
<function_decl 0x40293f50 strtol
    type <function_type 0x40293ee0
        type <integer_type 0x40166460 long int SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set 24 precision 32
            min <integer_cst 0x40179020 -2147483648>
            max <integer_cst 0x40179040 2147483647>
            pointer_to_this <pointer_type 0x4017d7e0>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 64 symtab 0 alias set -1
        arg-types <tree_list 0x40294ab4 value <pointer_type 0x40227690>
            chain <tree_list 0x40294ac8 value <pointer_type 0x4022ea10>
                chain <tree_list 0x40294adc value <integer_type 0x40166380 int>
                    chain <tree_list 0x40294af0
                        value <void_type 0x4017a770 void>>>>>
        pointer_to_this <pointer_type 0x402a7b60>>
    used nothrow public
    in_system_header external inline defer-output
        QI file /usr/include/stdlib.h line 301

    arguments <parm_decl 0x402a4070 __nptr
        type <pointer_type 0x40227690 type <integer_type 0x4017d700 char>
            unsigned SI
            size <integer_cst 0x40179540 constant 32>
            unit size <integer_cst 0x401795a0 constant 4>
            align 32 symtab 0 alias set -1>
        unsigned used in_system_header SI file /usr/include/stdlib.h line 299
        size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
        align 32 alias set -2
        context <function_decl 0x40293f50 strtol>
        result <pointer_type 0x40227690> initial <pointer_type 0x40227690>
        (reg/v/f:SI 59) arg-type <pointer_type 0x40227690>
        arg-type-as-written <pointer_type 0x40227690>
        incoming-rtl (mem/f:SI
            (reg/f:SI 53 virtual-incoming-args) [20 __nptr+0 S4 A32])
        chain <parm_decl 0x402a40e0 __endptr type <pointer_type 0x4022ea10>
            unsigned used in_system_header SI file /usr/include/stdlib.h line 299
                size <integer_cst 0x40179540 32>
                unit size <integer_cst 0x401795a0 4>
```

```
            align 32 alias set -2 context <function_decl 0x40293f50 strtol>
                result <pointer_type 0x4022ea10> initial <pointer_type 0x4022ea10>
            (reg/v/f:SI 60) arg-type <pointer_type 0x4022ea10>
                arg-type-as-written <pointer_type 0x4022ea10>
            incoming-rtl (mem/f:SI (plus:SI (reg/f:SI 53 virtual-incoming-args)
        (const_int 4 [0x4])) [30 __endptr+0 S4 A32])
        chain <parm_decl 0x402a4150 __base>>>
    result <result_decl 0x402a42a0 type <integer_type 0x40166460 long int>
        in_system_header regdecl SI file /usr/include/stdlib.h line 301
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
        align 32 context <function_decl 0x40293f50 strtol>
        (reg:SI 58)> initial <block 0x40290480>
    (mem:QI (symbol_ref:SI ("strtol")) [0 S1 A8])
    saved-insns 0x4033e300 chain <function_decl 0x40293d20 strtold>>
```

LABEL_DECL d 0

아직 정확한 설명은 없습니다.

```
<label_decl 0x40182770 test_label
    type <void_type 0x4016a770 void VOID
        align 8 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x4016a7e0>>
    VOID file <stdin> line 6
    align 1 context <function_decl 0x40182620 main>>
```

CONST_DECL d 0

아직 정확한 설명은 없습니다.

```
 <const_decl 0x401a2a80 PROCESSOR_I386
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40176f60 -2147483648>
        max <integer_cst 0x40176f80 2147483647>
        pointer_to_this <pointer_type 0x4017d620>>
    VOID file config/i386/i386.h line 400
    align 1 initial <integer_cst 0x40179620 0> chain <type_decl 0x401a2a10>>
 <const_decl 0x401d0af0 IX86_BUILTIN_CMPNGEPS
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40176f60 -2147483648>
        max <integer_cst 0x40176f80 2147483647>
        pointer_to_this <pointer_type 0x4017d620>>
    VOID file config/i386/i386.h line 2057
    align 1 initial <integer_cst 0x401cc700 17>
    chain <const_decl 0x401d0a80 IX86_BUILTIN_CMPNGTPS>>
```

TYPE_DECL d 0

아직 정확한 설명은 없습니다.

```
<type_decl 0x401a41c0
   type <enumeral_type 0x401a4150 reg_class VOID
       align 8 symtab 0 alias set -1 precision 0>
   VOID file config/i386/i386.h line 1210
   align 1 chain <var_decl 0x401a40e0 ix86_arch>>
<type_decl 0x401e1d90
   type <enumeral_type 0x401e1d20 fp_cw_mode VOID
       align 8 symtab 0 alias set -1 precision 0>
   VOID file config/i386/i386.h line 3159
   align 1 chain <var_decl 0x401e1cb0 ix86_compare_op1>>
<type_decl 0x4016e930 __builtin_va_list
   type <pointer_type 0x4016acb0 __builtin_va_list
       type <integer_type 0x40166230 char QI
           size <integer_cst 0x401633e0 constant 8>
           unit size <integer_cst 0x40163400 constant 1>
           align 8 symtab 0 alias set -1 precision 8
           min <integer_cst 0x401634a0 -128>
           max <integer_cst 0x401634c0 127>
           pointer_to_this <pointer_type 0x4016ac40>>
       unsigned SI
       size <integer_cst 0x40163b80 constant 32>
       unit size <integer_cst 0x40163be0 constant 4>
       align 32 symtab 0 alias set -1>
   VOID file <built-in> line 0
   align 1 chain <type_decl 0x4016e4d0 void>>
```

VAR_DECL d 0

아직 정확한 설명은 없습니다.

```
<var_decl 0x40182540 xxy_us_dummy
   type <integer_type 0x40166380 int SI
       size <integer_cst 0x40163540 constant 32>
       unit size <integer_cst 0x401635e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401635a0 -2147483648>
       max <integer_cst 0x401635c0 2147483647>
       pointer_to_this <pointer_type 0x4016e620>>
   SI file tmp-dum.c line 1
   size <integer_cst 0x40163540 32> unit size <integer_cst 0x401635e0 4>
   align 32>
<var_decl 0x4019e850 ix86_cost
   type <pointer_type 0x4019e7e0
       type <record_type 0x4019e770 processor_costs readonly tree_1 type_0 BLK
           size <integer_cst 0x4019d880 constant 1280>
           unit size <integer_cst 0x4019d860 constant 160>
           align 32 symtab 0 alias set -1 fields <field_decl 0x40191cb0 add>
           pointer_to_this <pointer_type 0x4019e7e0>>
       unsigned SI
```

```
            size <integer_cst 0x40179540 constant 32>
            unit size <integer_cst 0x401795a0 constant 4>
            align 32 symtab 0 alias set -1>
        unsigned SI file config/i386/i386.h line 95
        size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
        align 32>
    <var_decl 0x4019e930 x86_use_leave
        type <integer_type 0x4019c310 int readonly SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40176f60 -2147483648>
            max <integer_cst 0x40176f80 2147483647>
            pointer_to_this <pointer_type 0x4019c380>>
        SI file config/i386/i386.h line 213
        size <integer_cst 0x40176f00 32> unit size <integer_cst 0x40176fa0 4>
        align 32>
    <var_decl 0x402217e0 _IO_2_1_stdin_
        type <record_type 0x40221700 _IO_FILE_plus VOID
            align 8 symtab 0 alias set -1
            chain <type_decl 0x40221770>>
        in_system_header VOID file /usr/include/libio.h line 326
        align 8>
    <var_decl 0x402299a0 stdin
        type <pointer_type 0x40229930
            type <record_type 0x40202f50 FILE BLK
                size <integer_cst 0x4021f400 constant 1184>
                unit size <integer_cst 0x4021f3c0 constant 148>
                align 32 symtab 0 alias set -1 fields <field_decl 0x4021b850 _flags>
                pointer_to_this <pointer_type 0x40229930>>
            unsigned SI
            size <integer_cst 0x40179540 constant 32>
            unit size <integer_cst 0x401795a0 constant 4>
            align 32 symtab 0 alias set -1>
        unsigned in_system_header SI file include/stdio.h line 142
        size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
        align 32>
```

PARM_DECL d 0

아직 정확한 설명은 없습니다.

```
    <parm_decl 0x40208770
        type <pointer_type 0x40208700
            type <record_type 0x40208380 __gconv_step VOID
                align 8 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x40208700>
                chain <type_decl 0x402083f0>>
            unsigned SI
            size <integer_cst 0x40179540 constant 32>
            unit size <integer_cst 0x401795a0 constant 4>
            align 32 symtab 0 alias set -1>
        unsigned in_system_header SI file /usr/include/gconv.h line 70
```

```
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
 <parm_decl 0x40208850
    type <pointer_type 0x402087e0
        type <record_type 0x40208460 __gconv_step_data VOID
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x402087e0> chain <type_decl 0x402084d0>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1>
    unsigned in_system_header SI file /usr/include/gconv.h line 70
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
 <parm_decl 0x40208a80
    type <pointer_type 0x40208930
        type <integer_type 0x402088c0 unsigned char readonly unsigned QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x40176e20 0> max <integer_cst 0x40176e40 255>
            pointer_to_this <pointer_type 0x40208930>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402089a0>>
    unsigned in_system_header SI file /usr/include/gconv.h line 71
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
 <parm_decl 0x40215460
    type <pointer_type 0x40208700
        type <record_type 0x40208380 __gconv_step VOID
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x40208700> chain <type_decl 0x402083f0>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1>
    unsigned in_system_header SI file /usr/include/gconv.h line 80
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
```

RESULT_DECL d 0

아직 정확한 설명은 없습니다.

```
 <result_decl 0x40246620
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x40176f60 -2147483648>
```

```
        max <integer_cst 0x40176f80 2147483647>
        pointer_to_this <pointer_type 0x4017d620>>
    in_system_header SI file /usr/include/bits/stdio.h line 35
        size <integer_cst 0x40176f00 32>
        unit size <integer_cst 0x40176fa0 4>
    align 32>
<result_decl 0x40246850
    type <integer_type 0x40166380 int SI
        size <integer_cst 0x40176f00 constant 32>
        unit size <integer_cst 0x40176fa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32 '
        min <integer_cst 0x40176f60 -2147483648>
        max <integer_cst 0x40176f80 2147483647>
        pointer_to_this <pointer_type 0x4017d620>>
    in_system_header SI file /usr/include/bits/stdio.h line 42
    size <integer_cst 0x40176f00 32> unit size <integer_cst 0x40176fa0 4>
    align 32>
<result_decl 0x4030d2a0
    type <pointer_type 0x4030d150
        type <record_type 0x40309a10 dwarf_cie packed type_0 BLK
            size <integer_cst 0x401796c0 constant 96>
            unit size <integer_cst 0x40179700 constant 12>
            user align 32 symtab 0 alias set -1
            attributes <tree_list 0x40308de8
                purpose <identifier_node 0x4030b200 aligned>
                value <tree_list 0x40308dac
                    value <integer_cst 0x4030a060 constant 4>>
                chain <tree_list 0x40308dd4
                    purpose <identifier_node 0x40267fc0 packed>>>
            fields <field_decl 0x40309af0 length>
            pointer_to_this <pointer_type 0x4030d150>
            chain <type_decl 0x40309a80>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1>
    unsigned SI file unwind-dw2-fde.h line 152
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
<result_decl 0x4030d540
    type <pointer_type 0x4030d380
        type <record_type 0x4030d070 fde packed type_0 BLK
            size <integer_cst 0x401792c0 constant 64>
            unit size <integer_cst 0x401794e0 constant 8>
            user align 32 symtab 0 alias set -1
            attributes <tree_list 0x4030c03c
                purpose <identifier_node 0x4030b200 aligned>
                value <tree_list 0x4030c000
                    value <integer_cst 0x4030a0c0 constant 4>>
                chain <tree_list 0x4030c028
                    purpose <identifier_node 0x40267fc0 packed>>>
            fields <field_decl 0x40309d90 length>
```

```
            pointer_to_this <pointer_type 0x4030d380>>
        unsigned SI
        size <integer_cst 0x40179540 constant 32>
        unit size <integer_cst 0x401795a0 constant 4>
        align 32 symtab 0 alias set -1>
    unsigned SI file unwind-dw2-fde.h line 158
    size <integer_cst 0x40179540 32> unit size <integer_cst 0x401795a0 4>
    align 32>
```

FIELD_DECL d 0

아직 정확한 설명은 없습니다.

```
 <field_decl 0x4017af50 f
    type <array_type 0x4017ae70
        type <integer_type 0x40166930 unsigned SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40179360 0>
            max <integer_cst 0x40179380 4294967295>
            pointer_to_this <pointer_type 0x4017ae00>>
        BLK
        size <integer_cst 0x401797a0 constant 128>
        unit size <integer_cst 0x401797c0 constant 16>
        align 32 symtab 0 alias set -1
        domain <integer_type 0x4017ad90
            type <integer_type 0x4017a4d0 unsigned int>
            SI size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401795c0 0> max <integer_cst 0x40179740 3>>>
    BLK file <built-in> line 0
    size <integer_cst 0x401797a0 128> unit size <integer_cst 0x401797c0 16>
    align 32 offset_align 1 context <record_type 0x4017aee0>>
 <field_decl 0x4017b1c0 f
    type <array_type 0x4017b0e0
        type <integer_type 0x40166930 unsigned SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x40179360 0> max <integer_cst 0x40179380 4294967295>
            pointer_to_this <pointer_type 0x4017ae00>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 32 symtab 0 alias set -1
        domain <integer_type 0x4017b070 type <integer_type 0x4017a4d0 unsigned int>
            SI size <integer_cst 0x40176f00 32>
            unit size <integer_cst 0x40176fa0 4>
            align 32 symtab 0 alias set -1 precision 32
    min <integer_cst 0x401795c0 0> max <integer_cst 0x40179520 1>>>
```

```
    DI file <built-in> line 0
    size <integer_cst 0x401792c0 64> unit size <integer_cst 0x401794e0 8>
    align 32 offset_align 1 context <record_type 0x4017b150>>
<field_decl 0x4017b460 f
    type <array_type 0x4017b380
        type <integer_type 0x401668c0 unsigned HI
            size <integer_cst 0x40176e00 constant 16>
            unit size <integer_cst 0x40176ee0 constant 2>
            align 16 symtab 0 alias set -1 precision 16
            min <integer_cst 0x40179320 0> max <integer_cst 0x40179340 65535>
            pointer_to_this <pointer_type 0x4017b310>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 16 symtab 0 alias set -1
        domain <integer_type 0x4017ad90 type <integer_type 0x4017a4d0 unsigned int>
            SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401795c0 0> max <integer_cst 0x40179740 3>>>
    DI file <built-in> line 0
    size <integer_cst 0x401792c0 64> unit size <integer_cst 0x401794e0 8>
    align 16 offset_align 1 context <record_type 0x4017b3f0>>
<field_decl 0x4017b700 f
    type <array_type 0x4017b620
        type <integer_type 0x40166850 unsigned QI
            size <integer_cst 0x40176da0 constant 8>
            unit size <integer_cst 0x40176dc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401792e0 0> max <integer_cst 0x40179300 255>
            pointer_to_this <pointer_type 0x4017b5b0>>
        DI
        size <integer_cst 0x401792c0 constant 64>
        unit size <integer_cst 0x401794e0 constant 8>
        align 8 symtab 0 alias set -1
        domain <integer_type 0x4017b540 type <integer_type 0x4017a4d0 unsigned int>
            SI
            size <integer_cst 0x40176f00 constant 32>
            unit size <integer_cst 0x40176fa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401795c0 0> max <integer_cst 0x401798e0 7>>>
    DI file <built-in> line 0
    size <integer_cst 0x401792c0 64> unit size <integer_cst 0x401794e0 8>
    align 8 offset_align 1 context <record_type 0x4017b690>>
```

NAMESPACE_DECL d 0

Namespace 선언. Namespace 들은 이름들의 체계를 제공하는데, 다른 _DECL 들의 DECL_CONTEXT 내에서 나타난다.

C 예제가 존재하지 않습니다.

저장소로의 reference.

COMPONENT_REF r 2

값은 구조체 혹은 공용체 component 이다. Operand 0 은 구조체 혹은 공용체 (표현식) 이다. Operand 1 은 field (Type FIELD_DECL 의 node) 이다.

```
<component_ref 0x402a08a0
    type <pointer_type 0x401eec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128>
            max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402979a0>>
    arg 0 <indirect_ref 0x402aef8c
        type <record_type 0x4026be00 _IO_FILE type_0 BLK
            size <integer_cst 0x40288400 constant 1184>
            unit size <integer_cst 0x402883c0 constant 148>
            align 32 symtab 0 alias set -1
            fields <field_decl 0x40284850 _flags
                type <integer_type 0x401da380 int SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eaf60 -2147483648>
                    max <integer_cst 0x401eaf80 2147483647>
                    pointer_to_this <pointer_type 0x401f1620>>
                in_system_header SI
                file /usr/include/libio.h line 262
                size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 32 offset_align 128
                offset <integer_cst 0x401ed5c0 constant 0>
                bit offset <integer_cst 0x401ed680 constant 0>
                context <record_type 0x4026be00 _IO_FILE>
                arguments <integer_cst 0x401ed5c0 0>
                chain <field_decl 0x402848c0 _IO_read_ptr>>
            pointer_to_this <pointer_type 0x40284460>
            chain <type_decl 0x4026be70>>
        arg 0 <parm_decl 0x402af930 __fp
            type <pointer_type 0x40292930
                type <record_type 0x4026bf50 FILE BLK
                size <integer_cst 0x40288400 1184>
                unit size <integer_cst 0x402883c0 148>
                    align 32 symtab 0 alias set -1
                    fields <field_decl 0x40284850 _flags>
                    pointer_to_this <pointer_type 0x40292930>>
                unsigned SI size <integer_cst 0x401ed540 32>
```

```
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1>
            unsigned used in_system_header SI
            file /usr/include/bits/stdio.h line 50
            size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32
            context <function_decl 0x4029fb60 getc_unlocked>
            result <pointer_type 0x40292930>
            initial <pointer_type 0x40292930>
            arg-type <pointer_type 0x40292930>
            arg-type-as-written <pointer_type 0x40292930>>>
   arg 1 <field_decl 0x402848c0 _IO_read_ptr type <pointer_type 0x401eec40>
       unsigned in_system_header SI
       file /usr/include/libio.h line 267
       size <integer_cst 0x401ed540 32>
       unit size <integer_cst 0x401ed5a0 4>
       align 32 offset_align 128
       offset <integer_cst 0x401ed5c0 0>
       bit offset <integer_cst 0x401eaf00 32>
       context <record_type 0x4026be00 _IO_FILE>
       arguments <integer_cst 0x401ed5c0 0>
       chain <field_decl 0x40284930 _IO_read_end type <pointer_type 0x401eec40>
           unsigned in_system_header SI
           file /usr/include/libio.h line 268
           size <integer_cst 0x401ed540 32>
           unit size <integer_cst 0x401ed5a0 4>
           align 32 offset_align 128 offset <integer_cst 0x401ed5c0 0>
           bit offset <integer_cst 0x401ed2c0 constant 64>
           context <record_type 0x4026be00 _IO_FILE>
           arguments <integer_cst 0x401ed5c0 0>
           chain <field_decl 0x402849a0 _IO_read_base>>>>
<component_ref 0x40373120
   type <integer_type 0x40372620 sword SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eaf60 -2147483648>
       max <integer_cst 0x401eaf80 2147483647>>
   arg 0 <indirect_ref 0x4037530c
       type <record_type 0x4036e150 dwarf_fde packed type_0 BLK
           size <integer_cst 0x401ed2c0 constant 64>
           unit size <integer_cst 0x401ed4e0 constant 8>
           user align 32 symtab 0 alias set -1
           attributes <tree_list 0x4037503c
               purpose <identifier_node 0x40374200 aligned>
               value <tree_list 0x40375000
                   value <integer_cst 0x403730c0 constant 4>>
               chain <tree_list 0x40375028
                   purpose <identifier_node 0x402d0fc0 packed>>>
           fields <field_decl 0x40372d90 length
               type <integer_type 0x40372700 uword unsigned SI
```

```
                    size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eafe0 0>
                    max <integer_cst 0x401ed000 4294967295>>
                unsigned packed SI file unwind-dw2-fde.h line 141
                size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 8 offset_align 128
                offset <integer_cst 0x401ed5c0 constant 0>
                bit offset <integer_cst 0x401ed680 constant 0>
                context <record_type 0x4036e150 dwarf_fde>
                arguments <integer_cst 0x401ed5c0 0>
                chain <field_decl 0x40372e00 CIE_delta>>
            pointer_to_this <pointer_type 0x4036e230> chain <type_decl 0x4036e1c0>>
        arg 0 <parm_decl 0x403760e0 f
            type <pointer_type 0x4036e230 type <record_type 0x4036e150 dwarf_fde>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x4036e310>>
            unsigned used SI file unwind-dw2-fde.h line 151
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x40376230 get_cie>
            result <pointer_type 0x4036e230>
            initial <pointer_type 0x4036e230>
            arg-type <pointer_type 0x4036e230>
            arg-type-as-written <pointer_type 0x4036e230>>>
    arg 1 <field_decl 0x40372e00 CIE_delta type <integer_type 0x40372620 sword>
        packed SI file unwind-dw2-fde.h line 142
        size <integer_cst 0x401eaf00 32>
        unit size <integer_cst 0x401eafa0 4>
        align 8 offset_align 128 offset <integer_cst 0x401ed5c0 0>
        bit offset <integer_cst 0x401eaf00 32>
        context <record_type 0x4036e150 dwarf_fde>
        arguments <integer_cst 0x401ed5c0 0>
        chain <field_decl 0x40372f50 pc_begin
            type <array_type 0x40372ee0
                type <integer_type 0x401da1c0 unsigned char unsigned QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set -1 precision 8
                    min <integer_cst 0x401eae20 0>
                    max <integer_cst 0x401eae40 255>
                    pointer_to_this <pointer_type 0x40271af0>>
                BLK
                align 8 symtab 0 alias set -1
                domain <integer_type 0x40372e70
                    type <integer_type 0x401ee4d0 unsigned int
                        unsigned sizetype SI
                        size <integer_cst 0x401eaf00 32>
```

```
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eafe0 0>
                    max <integer_cst 0x401ed000 4294967295>>
                SI size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401ed5c0 0>>>
            packed BLK file unwind-dw2-fde.h line 143
            align 8 offset_align 128 offset <integer_cst 0x401ed5c0 0>
            bit offset <integer_cst 0x401ed2c0 64>
            context <record_type 0x4036e150 dwarf_fde>
            arguments <integer_cst 0x401ed5c0 0>>>>
```

BIT_FIELD_REF r 3

Onject 내부의 bit 들의 group 으로의 reference. COMPONENT_REF 와 비슷하지만 위치가 FIELD_DECL 를 통하는 것 보다 분명하게 주어진다. Operand 0 는 구조체 혹은 공용체 (표현식) 이다.
Operand 1 는 참조되는 bit 들의 갯수를 주는 tree;
Operand 2 는 처음 참조되는 bit 의 위치를 주는 tree.
Field 는 signed 혹은 unsigned field 일 수 있다; TREE_UNSIGNED 가 어떤 것인지 가르킨다.

적당한 C 예제를 추출하지 못하였습니다.

INDIRECT_REF r 1

C unary '*' 혹은 Pascal '^'. 하나의 operand 는 pointer 에 대한 expression 이다.

```
 <indirect_ref 0x402d65b4
    type <integer_type 0x401da230 char QI
        size <integer_cst 0x401eada0 constant 8>
        unit size <integer_cst 0x401eadc0 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
        pointer_to_this <pointer_type 0x401eec40>>
    readonly
    arg 0 <plus_expr 0x402d53a0
        type <pointer_type 0x401f1770
            type <integer_type 0x401f1700 char readonly QI
            size <integer_cst 0x401eada0 8>
            unit size <integer_cst 0x401eadc0 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128>
                max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401f1770>>
            unsigned SI
            size <integer_cst 0x401ed540 constant 32>
            unit size <integer_cst 0x401ed5a0 constant 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x4027ee00>>
        arg 0 <parm_decl 0x402d3b60 __s type <pointer_type 0x401f1770>
            unsigned used in_system_header SI
            file /usr/include/bits/string2.h line 930
```

```
            size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402d3af0 __strcspn_c2>
            result <pointer_type 0x401f1770>
            initial <pointer_type 0x401f1770>
            arg-type <pointer_type 0x401f1770>
            arg-type-as-written <pointer_type 0x401f1770>
            chain <parm_decl 0x402d3bd0 __reject1>>
        arg 1 <convert_expr 0x402d65a0 type <pointer_type 0x401f1770>
            arg 0 <non_lvalue_expr 0x402d658c
                type <integer_type 0x402520e0 size_t unsigned SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eafe0 0>
                    max <integer_cst 0x401ed000 4294967295>
                    pointer_to_this <pointer_type 0x40271c40>>
                arg 0 <var_decl 0x402d3e00 __result
                    type <integer_type 0x402520e0 size_t>
                    unsigned used in_system_header common regdecl SI
                    file /usr/include/bits/string2.h line 932
                    size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 context <function_decl 0x402d3af0
                        __strcspn_c2>
                    initial <integer_cst 0x402d5320 0>>>>>>
<indirect_ref 0x402db6a4
   type <integer_type 0x401da230 char QI
       size <integer_cst 0x401eada0 constant 8>
       unit size <integer_cst 0x401eadc0 constant 1>
       align 8 symtab 0 alias set -1 precision 8
       min <integer_cst 0x401eae60 -128>
       max <integer_cst 0x401eae80 127>
       pointer_to_this <pointer_type 0x401eec40>>
   readonly
   arg 0 <plus_expr 0x402d5f20
       type <pointer_type 0x401f1770
           type <integer_type 0x401f1700 char readonly QI
           size <integer_cst 0x401eada0 8>
           unit size <integer_cst 0x401eadc0 1>
               align 8 symtab 0 alias set -1 precision 8
               min <integer_cst 0x401eae60 -128>
               max <integer_cst 0x401eae80 127>
               pointer_to_this <pointer_type 0x401f1770>>
           unsigned SI
           size <integer_cst 0x401ed540 constant 32>
           unit size <integer_cst 0x401ed5a0 constant 4>
           align 32 symtab 0 alias set -1
           pointer_to_this <pointer_type 0x4027ee00>>
       arg 0 <parm_decl 0x402dc150 __s type <pointer_type 0x401f1770>
           unsigned used in_system_header SI
           file /usr/include/bits/string2.h line 1000
```

```
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402dc0e0 __strspn_c3>
                result <pointer_type 0x401f1770>
                initial <pointer_type 0x401f1770>
                arg-type <pointer_type 0x401f1770>
                arg-type-as-written <pointer_type 0x401f1770>
                chain <parm_decl 0x402dc1c0 __accept1>>
            arg 1 <convert_expr 0x402db690 type <pointer_type 0x401f1770>
                arg 0 <non_lvalue_expr 0x402db67c
                    type <integer_type 0x402520e0 size_t unsigned SI
                        size <integer_cst 0x401eaf00 constant 32>
                        unit size <integer_cst 0x401eafa0 constant 4>
                        align 32 symtab 0 alias set -1 precision 32
                        min <integer_cst 0x401eafe0 0>
                        max <integer_cst 0x401ed000 4294967295>
                        pointer_to_this <pointer_type 0x40271c40>>
                    arg 0 <var_decl 0x402dc460 __result
                        type <integer_type 0x402520e0 size_t>
                        unsigned used in_system_header common regdecl SI
                        file /usr/include/bits/string2.h line 1002
                        size <integer_cst 0x401eaf00 32>
                        unit size <integer_cst 0x401eafa0 4>
                        align 32 context <function_decl 0x402dc0e0
                            __strspn_c3> initial <integer_cst 0x402d5e20 0>>>>>>
    <indirect_ref 0x402f1550
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128>
            max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        arg 0 <plus_expr 0x402ddce0
            type <pointer_type 0x401eec40 type <integer_type 0x401da230 char>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x402979a0>>
            arg 0 <parm_decl 0x402f01c0 __s type <pointer_type 0x401eec40>
                unsigned used in_system_header SI
                file /usr/include/bits/string2.h line 1085
                size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402f0150 __strtok_r_1c>
                result <pointer_type 0x401eec40>
                initial <pointer_type 0x401eec40>
                arg-type <pointer_type 0x401eec40>
                arg-type-as-written <pointer_type 0x401eec40>
                chain <parm_decl 0x402f0230 __sep>>
            arg 1 <integer_cst 0x402ddcc0 constant 4294967295>>>
```

BUFFER_REF r 1

파일에 관한 Pascal '^'. 하나의 operand 는 file 에 대한 expression 이다.

C 예제가 존재하지 않습니다.

ARRAY_REF r 2

배열 indexing.
Operand 0 은 배열; operand 1 은 (single) 배열 index 이다.

```
 <array_ref 0x4037bd80
    type <pointer_type 0x401ee7e0
        type <void_type 0x401ee770 void VOID
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x401ee7e0>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set 7
        pointer_to_this <pointer_type 0x402801c0>>
    arg 0 <var_decl 0x40378a80 __JCR_LIST__
        type <array_type 0x40378a10 type <pointer_type 0x401ee7e0>
            BLK
            size <integer_cst 0x401ed680 constant 0>
            unit size <integer_cst 0x401ed5c0 constant 0>
            align 32 symtab 0 alias set 6
            domain <integer_type 0x40378af0
                type <integer_type 0x401ee4d0 unsigned int unsigned sizetype SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eafe0 0>
                    max <integer_cst 0x401ed000 4294967295>>
                SI size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401ed5c0 0>
                max <integer_cst 0x403738e0 -1>>>
        addressable asm_written used static common BLK
        file crtstuff.c line 203
        size <integer_cst 0x401ed680 0>
        unit size <integer_cst 0x401ed5c0 0>
        user align 32 attributes <tree_list 0x40377b54>
        initial <constructor 0x40373a60>
        (mem/s:BLK (symbol_ref:SI ("__JCR_LIST__")) [6 __JCR_LIST__+0 S0 A32])
        chain <var_decl 0x40378850 __EH_FRAME_BEGIN__
            type <array_type 0x403787e0
                type <integer_type 0x401da230 char QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set 5 precision 8
                    min <integer_cst 0x401eae60 -128>
                    max <integer_cst 0x401eae80 127>
```

```
                    pointer_to_this <pointer_type 0x401eec40>>
                BLK size <integer_cst 0x401ed680 0>
                unit size <integer_cst 0x401ed5c0 0>
                align 8 symtab 0 alias set 4
                domain <integer_type 0x403788c0
                    type <integer_type 0x401ee4d0 unsigned int>
                    SI size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401ed5c0 0>
                    max <integer_cst 0x403738e0 -1>>>
            addressable asm_written used static common BLK
            file crtstuff.c line 195 size <integer_cst 0x401ed680 0>
            unit size <integer_cst 0x401ed5c0 0>
            user align 32 attributes <tree_list 0x4037799c>
            initial <constructor 0x40373920>
            (mem/s:BLK (symbol_ref:SI ("__EH_FRAME_BEGIN__"))
                [4 __EH_FRAME_BEGIN__+0 S0 A32])
                chain <var_decl 0x40378690 __DTOR_LIST__>>>
    arg 1 <integer_cst 0x401ed620
        type <integer_type 0x401da380 int> constant 0>>
<array_ref 0x4037bd80
    type <pointer_type 0x401ee7e0
        type <void_type 0x401ee770 void VOID
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x401ee7e0>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set 7
        pointer_to_this <pointer_type 0x402811c0>>
    arg 0 <var_decl 0x40378a80 __JCR_LIST__
        type <array_type 0x40378a10 type <pointer_type 0x401ee7e0>
            BLK
            size <integer_cst 0x401ed680 constant 0>
            unit size <integer_cst 0x401ed5c0 constant 0>
            align 32 symtab 0 alias set 6
            domain <integer_type 0x40378af0
                type <integer_type 0x401ee4d0 unsigned int unsigned sizetype SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eafe0 0>
                    max <integer_cst 0x401ed000 4294967295>>
                SI size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401ed5c0 0> max <integer_cst 0x403738e0 -1>>>
        addressable asm_written used static common BLK
        file crtstuff.c line 203 size <integer_cst 0x401ed680 0>
        unit size <integer_cst 0x401ed5c0 0>
        user align 32 attributes <tree_list 0x40377b54>
```

```
        initial <constructor 0x40373a60>
        (mem/s:BLK (symbol_ref:SI ("__JCR_LIST__")) [6 __JCR_LIST__+0 S0 A32])
        chain <var_decl 0x40378850 __EH_FRAME_BEGIN__
            type <array_type 0x403787e0
                type <integer_type 0x401da230 char QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set 5 precision 8
                    min <integer_cst 0x401eae60 -128>
                    max <integer_cst 0x401eae80 127>
                    pointer_to_this <pointer_type 0x401eec40>>
                BLK size <integer_cst 0x401ed680 0>
                unit size <integer_cst 0x401ed5c0 0>
                align 8 symtab 0 alias set 4
                domain <integer_type 0x403788c0
                    type <integer_type 0x401ee4d0 unsigned int>
                    SI size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401ed5c0 0>
                    max <integer_cst 0x403738e0 -1>>>
            addressable asm_written used static common BLK
            file crtstuff.c line 195 size <integer_cst 0x401ed680 0>
            unit size <integer_cst 0x401ed5c0 0>
            user align 32 attributes <tree_list 0x4037799c>
            initial <constructor 0x40373920>
            (mem/s:BLK (symbol_ref:SI ("__EH_FRAME_BEGIN__"))
                [4 __EH_FRAME_BEGIN__+0 S0 A32])
            chain <var_decl 0x40378690 __DTOR_LIST__>>>
    arg 1 <integer_cst 0x401ed620 type <integer_type 0x401da380 int> constant 0>>
```

ARRAY_RANGE_REF r 2

위와 비슷하지만 결과가 배열의 범위("slice") 이다. 결과 배열의 시작 index 는 operand 1 로부터 얻어지며 범위의 크기는 표현식의 type 으로부터 얻어진다.

적당한 C 예제를 추출하지 못하였습니다.

VTABLE_REF r 3

Vtable indexing. vtable garbage collection 을 위한 정보를 내는데 유용한 data 를 운반한다.
Operand 0: 하나의 array_ref (혹은 동일한 표현식)
Operand 1: vtable base (반드시 var_decl 여야 함.)
Operand 2: vtable 내의 index (반드시 integer_cst 여야 함).

적당한 C 예제를 추출하지 못하였습니다.

CONSTRUCTOR e 2

Constructor: 지정된 component 들로 구정된 집합값을 반환한다. C 에서, 이것은 구조체와 배열 초기화자에만 사용된다. 또한 Chill (와 잠재적으로는 Pascal) 에서는 SET_TYPE 에 대해서도 사용된다. 첫번째 "operand" 는 실제로는 constant constructor 들에 대한 RTL 로의 pointer 이다. 두번째 operand 는 TREE_LIST node 들의 chain 으로 된 component 값들의 list 이다.

ARRAY_TYPE 용:

각 node 의 TREE_PURPOSE 는 대응하는 index 이다. 만약 TREE_PURPOSE 가 RANGE_EXPR 이면, 그것은 많은 node 들을 위한 속기(short-hand) 인 범위내의 각 index 를 위한 것이다. (만약 대응하는 TREE_VALUE 가 다른-영향을 가지고 있다면, 그들은 각 element 에 대해 한번 평가될 것이다. 만약 다른-영향을 오직 한번만 평가하길 원한다면 SAVE_EXPR 내의 값을 감싸라.)

RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE 용:
각 node 의 TREE_PURPOSE 는 FIELD_DECL 이다.

SET_TYPE 용:
TREE_VALUE 는 집합내에서 참인 값 (index) 이다. 만약 TREE_PURPOSE 가 NULL 이 아니면, 참 (true) 값의 범위 중 lower limit 를 지정한다. list 가 아닌 element 들은 거짓 (false) 이다. (집합에 포함안되는).

```
<constructor 0x40262f20
   type <array_type 0x402755b0
       type <integer_type 0x401da380 int SI
           size <integer_cst 0x401d7540 constant 32>
           unit size <integer_cst 0x401d75e0 constant 4>
           align 32 symtab 0 alias set -1 precision 32
           min <integer_cst 0x401d75a0 -2147483648>
           max <integer_cst 0x401d75c0 2147483647>
           pointer_to_this <pointer_type 0x401e2620>>
       VOID
       align 8 symtab 0 alias set -1>

   arg 1 <tree_list 0x40274488
       purpose <integer_cst 0x40262e60 constant 0>
       value <integer_cst 0x401d7c80 constant 1>
       chain <tree_list 0x4027449c
           purpose <integer_cst 0x40262e80 constant 1>
           value <integer_cst 0x401d7c60 constant 0>
           chain <tree_list 0x402744b0
               purpose <integer_cst 0x40262ec0 constant 2>
               value <integer_cst 0x40262ea0 constant 703>
               chain <tree_list 0x402744d8
                   purpose <integer_cst 0x40262f00 constant 3>
                   value <integer_cst 0x40262ee0 constant -1>>>>>>
```

표현식 type 들은 대부분 직설적인데, DEFTREECODE 의 네번째 인자가 얼마나 많은 operand 들이 있는지를 말해준다. 만약 다른 것이 지정되지 않았다면, operand 들은 표현식이고 모든 operand 들과 표현식들의 type 들은 반드시 모두 같아야 한다.

COMPOUND_EXPR e 2

계산할 두개의 표현식을 담고 있다, 하나는 다른것을 다른다. 처음 값은 무시됩니다. 두번째 값은 사용됩니다. 첫번째 표현식의 type 은 다른 type 과 동의할 필요는 없다

```
chain <compound_stmt 0x4039749c tree_2
    arg 0 <scope_stmt 0x403974b0 tree_0
        chain <expr_stmt 0x403974d8 tree_1
```

```
            arg 0 <modify_expr 0x403a8be0 type <real_type 0x401ee9a0 double>
                side-effects arg 0 <var_decl 0x4037ad20>
                arg 1 <call_expr 0x403a8c00 type <real_type 0x401ee9a0 double>
                    side-effects
                    arg 0 <addr_expr 0x403974ec
                        type <pointer_type 0x4030d000
                            type <function_type 0x40308d90
                                type <real_type 0x401ee9a0 double>
                                DI size <integer_cst 0x401ed2c0 64>
                                unit size <integer_cst 0x401ed4e0 8>
                                align 64 symtab 0 alias set -1
                                arg-types <tree_list 0x4030a03c
                                    value <pointer_type 0x40290690>
                                    chain <tree_list 0x4030a050
                                        value <pointer_type 0x40297a10>
                                        chain <tree_list 0x4030a064
                                            value <integer_type 0x401da380>
                                            chain <tree_list 0x4030a078>>>>
                                pointer_to_this <pointer_type 0x4030d000>>
                            unsigned SI size <integer_cst 0x401ed540 32>
                            unit size <integer_cst 0x401ed5a0 4>
                            align 32 symtab 0 alias set -1>
                        arg 0 <function_decl 0x40308e00 __strtod_internal
                            type <function_type 0x40308d90>
                            addressable used public in_system_header
                            common external defer-output QI
                            file /usr/include/stdlib.h line 252
                            (mem:QI (symbol_ref:SI ("__strtod_internal"))
                                [0 S1 A8])
                            chain <function_decl 0x40308bd0 __strtold_l>>>
                    arg 1 <tree_list 0x40397500 value <var_decl 0x4037abd0 __nptr>
                            chain <tree_list 0x40397514
                                value <var_decl 0x4037ac40 __endptr>
                            chain <tree_list 0x40397528
                                value <integer_cst 0x403a8c20 constant 0>>>>>>
                    chain <goto_stmt 0x4039753c tree_0
                        arg 0 <label_decl 0x4037acb0 VOID
                            file /usr/include/stdlib.h line 295
                            align 1 context <function_decl 0x402fc3f0 atof>>
                        chain <scope_stmt 0x40397550>>>>
chain <scope_stmt 0x40397564 tree_3 arg 0 <block 0x403a7340>
    chain <label_stmt 0x40397578 arg 0 <label_decl 0x4037acb0>
        chain <expr_stmt 0x40397474 addressable
            arg 0 <var_decl 0x4037ad20>>>>>>>>>>
                chain <compound_stmt 0x4037fe9c tree_2
                    arg 0 <scope_stmt 0x4037feb0 tree_0
                        chain <expr_stmt 0x4037fed8 tree_1
                            arg 0 <modify_expr 0x4039fc60
                                type <integer_type 0x401da540 long long int>
                                side-effects
                                arg 0 <var_decl 0x4039ba10>
                                arg 1 <call_expr 0x4039fc80>>
```

```
                        chain <goto_stmt 0x4037ff50 tree_0
                            arg 0 <label_decl 0x4039b9a0 VOID
                                file /usr/include/stdlib.h line 343
                                align 1
                                context <function_decl 0x402fc7e0 atoll>>
                            chain <scope_stmt 0x4037ff64>>>>
```

MODIFY_EXPR e 2

Assignment expression. Operand 0 는 설정할 것이고, 1 은 새값이다.

```
<modify_expr 0x402a0cc0
   type <integer_type 0x401da230 char QI
       size <integer_cst 0x401eada0 constant 8>
       unit size <integer_cst 0x401eadc0 constant 1>
       align 8 symtab 0 alias set -1 precision 8
       min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
       pointer_to_this <pointer_type 0x401eec40>>
   side-effects
   arg 0 <indirect_ref 0x402b0b2c type <integer_type 0x401da230 char>
       side-effects
       arg 0 <postincrement_expr 0x402a0ca0
           type <pointer_type 0x401eec40 type <integer_type 0x401da230 char>
               unsigned SI
               size <integer_cst 0x401ed540 constant 32>
               unit size <integer_cst 0x401ed5a0 constant 4>
               align 32 symtab 0 alias set -1
               pointer_to_this <pointer_type 0x402979a0>>
           side-effects
           arg 0 <component_ref 0x402a0c60 type <pointer_type 0x401eec40>
               arg 0 <indirect_ref 0x402b0af0
                   type <record_type 0x4026be00 _IO_FILE type_0 BLK
                       size <integer_cst 0x40288400 constant 1184>
                       unit size <integer_cst 0x402883c0 constant 148>
                       align 32 symtab 0 alias set -1
                       fields <field_decl 0x40284850 _flags
                           type <integer_type 0x401da380 int SI
                               size <integer_cst 0x401eaf00 constant 32>
                               unit size <integer_cst 0x401eafa0 constant 4>
                               align 32 symtab 0 alias set -1 precision 32
                               min <integer_cst 0x401eaf60 -2147483648>
                               max <integer_cst 0x401eaf80 2147483647>
                               pointer_to_this <pointer_type 0x401f1620>>
                           in_system_header SI
                           file /usr/include/libio.h line 262
                           size <integer_cst 0x401eaf00 32>
                           unit size <integer_cst 0x401eafa0 4>
                           align 32 offset_align 128
                           offset <integer_cst 0x401ed5c0 constant 0>
                           bit offset <integer_cst 0x401ed680 constant 0>
                           context <record_type 0x4026be00 _IO_FILE>
                           arguments <integer_cst 0x401ed5c0 0>
                           chain <field_decl 0x402848c0 _IO_read_ptr>>
```

```
                        pointer_to_this <pointer_type 0x40284460>
                        chain <type_decl 0x4026be70>>
                    arg 0 <parm_decl 0x402aff50 __stream
                        type <pointer_type 0x40292930
                            type <record_type 0x4026bf50 FILE BLK
                                size <integer_cst 0x40288400 1184>
                                unit size <integer_cst 0x402883c0 148>
                                align 32 symtab 0 alias set -1
                                fields <field_decl 0x40284850 _flags>
                                pointer_to_this <pointer_type 0x40292930>>
                            unsigned SI size <integer_cst 0x401ed540 32>
                            unit size <integer_cst 0x401ed5a0 4>
                            align 32 symtab 0 alias set -1>
                        unsigned used in_system_header SI
                        file /usr/include/bits/stdio.h line 75
                        size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 context <function_decl 0x402a24d0 fputc_unlocked>
                        result <pointer_type 0x40292930>
                        initial <pointer_type 0x40292930>
                        arg-type <pointer_type 0x40292930>
                        arg-type-as-written <pointer_type 0x40292930>>>
                arg 1 <field_decl 0x40284a80 _IO_write_ptr
                    type <pointer_type 0x401eec40>
                    unsigned in_system_header SI
                    file /usr/include/libio.h line 271
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 offset_align 128
                    offset <integer_cst 0x401ed7c0 constant 16>
                    bit offset <integer_cst 0x401ed540 32>
                    context <record_type 0x4026be00 _IO_FILE>
                    arguments <integer_cst 0x401ed7c0 16>
                    chain <field_decl 0x40284af0 _IO_write_end>>>
            arg 1 <integer_cst 0x402a0c80 constant 1>>>
    arg 1 <nop_expr 0x402b0b54 type <integer_type 0x401da230 char>
        arg 0 <parm_decl 0x402afee0 __c type <integer_type 0x401da380 int>
            used in_system_header SI file /usr/include/bits/stdio.h line 75
            size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402a24d0 fputc_unlocked>
            result <integer_type 0x401da380 int>
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>
            chain <parm_decl 0x402aff50 __stream>>>>
<modify_expr 0x402f2140
    type <integer_type 0x401da230 char QI
        size <integer_cst 0x401eada0 constant 8>
        unit size <integer_cst 0x401eadc0 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401eae60 -128>
        max <integer_cst 0x401eae80 127>
```

```
        pointer_to_this <pointer_type 0x401eec40>>
    side-effects
    arg 0 <indirect_ref 0x402f30a0 type <integer_type 0x401da230 char>
        side-effects
        arg 0 <postincrement_expr 0x402f20c0
            type <pointer_type 0x401eec40 type <integer_type 0x401da230 char>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x402979a0>>
            side-effects
            arg 0 <indirect_ref 0x402f3064 type <pointer_type 0x401eec40>
                arg 0 <parm_decl 0x402f0850 __s
                    type <pointer_type 0x402979a0 type <pointer_type 0x401eec40>
                        unsigned SI size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 symtab 0 alias set -1>
                    unsigned used in_system_header SI
                    file /usr/include/bits/string2.h line 1135
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x402f07e0 __strsep_1c>
                    result <pointer_type 0x402979a0>
                    initial <pointer_type 0x402979a0>
                    arg-type <pointer_type 0x402979a0>
                    arg-type-as-written <pointer_type 0x402979a0>
                    chain <parm_decl 0x402f08c0 __reject>>>
            arg 1 <integer_cst 0x402f20a0 constant 1>>>
    arg 1 <integer_cst 0x402f2120 type <integer_type 0x401da230 char> constant 0>>
```

INIT_EXPR e 2

초기화 expression. Operand 0 는 초기화할 값이고; Operand 1 는 초기화자 이다.

이 TREE node 를 위한 적당한 예제를 찾지 못하였습니다. 적당한 예제를 찾으신 분은 저에게 메일로 보내주시기 바랍니다.

TARGET_EXPR e 4

TARGET_EXPR 용으로써, operand 0 는 초기화의 target 이고, operand 1 는 target 에 대한 초기화자이고,
operand 2 는 만약 필요할시 이 node 를 위한 cleanup 이고,
operand 3 는 이 node 가 확장된 후 저장된 초기화자이다. 이것으로 우리는 나중에 tree 를 재-확장할 수 있다.

C 에서 사용되지 않는 TREE node 입니다.

COND_EXPR e 3

Conditional expression (C 에서 ... ? ... : ...).
Operand 0 는 조건문.
Operand 1 는 then-value 이다.

Operand 2 는 else-value 이다.
Operand 0 는 아무런 type 일것이다.
Operand 1 는 그것이 조건없이 예외상황으로 던져저 버리지 않는다면, 그러한 상황일 때는 VOID_TYPE 이여야 하겠지만, 않을 경우 반드시 전체 표현식과 같은 type 을 가져야한다. 같은 제약조건이 operand 2 에도 적용된다.

```
<cond_expr 0x402a0980
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eaf60 -2147483648>
       max <integer_cst 0x401eaf80 2147483647>
       pointer_to_this <pointer_type 0x401f1620>>
   side-effects
   arg 0 <ge_expr 0x402a08e0 type <integer_type 0x401da380 int>
       arg 0 <component_ref 0x402a08a0
           type <pointer_type 0x401eec40
               type <integer_type 0x401da230 char QI
                   size <integer_cst 0x401eada0 constant 8>
                   unit size <integer_cst 0x401eadc0 constant 1>
                   align 8 symtab 0 alias set -1 precision 8
                   min <integer_cst 0x401eae60 -128>
                   max <integer_cst 0x401eae80 127>
                   pointer_to_this <pointer_type 0x401eec40>>
               unsigned SI
               size <integer_cst 0x401ed540 constant 32>
               unit size <integer_cst 0x401ed5a0 constant 4>
               align 32 symtab 0 alias set -1
               pointer_to_this <pointer_type 0x402979a0>>

           arg 0 <indirect_ref 0x402aef8c
               type <record_type 0x4026be00 _IO_FILE type_0 BLK
                   size <integer_cst 0x40288400 constant 1184>
                   unit size <integer_cst 0x402883c0 constant 148>
                   align 32 symtab 0 alias set -1
                   fields <field_decl 0x40284850 _flags
                       type <integer_type 0x401da380 int>
                       in_system_header SI
                       file /usr/include/libio.h line 262
                       size <integer_cst 0x401eaf00 32>
                       unit size <integer_cst 0x401eafa0 4>
                       align 32 offset_align 128
                       offset <integer_cst 0x401ed5c0 constant 0>
                       bit offset <integer_cst 0x401ed680 constant 0>
                       context <record_type 0x4026be00 _IO_FILE>
                       arguments <integer_cst 0x401ed5c0 0>
                       chain <field_decl 0x402848c0 _IO_read_ptr>>
                   pointer_to_this <pointer_type 0x40284460>
                   chain <type_decl 0x4026be70>>
               arg 0 <parm_decl 0x402af930 __fp
                   type <pointer_type 0x40292930
```

```
                        type <record_type 0x4026bf50 FILE BLK
                        size <integer_cst 0x40288400 1184>
                        unit size <integer_cst 0x402883c0 148>
                            align 32 symtab 0 alias set -1
                            fields <field_decl 0x40284850 _flags>
                            pointer_to_this <pointer_type 0x40292930>>
                        unsigned SI size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 symtab 0 alias set -1>
                    unsigned used in_system_header SI
                    file /usr/include/bits/stdio.h line 50
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x4029fb60 getc_unlocked>
                    result <pointer_type 0x40292930>
                    initial <pointer_type 0x40292930>
                    arg-type <pointer_type 0x40292930>
                    arg-type-as-written <pointer_type 0x40292930>>>
            arg 1 <field_decl 0x402848c0 _IO_read_ptr
                type <pointer_type 0x401eec40>
                unsigned in_system_header SI
                file /usr/include/libio.h line 267
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 offset_align 128
                offset <integer_cst 0x401ed5c0 0>
                bit offset <integer_cst 0x401eaf00 32>
                context <record_type 0x4026be00 _IO_FILE>
                arguments <integer_cst 0x401ed5c0 0>
                chain <field_decl 0x40284930 _IO_read_end>>>
        arg 1 <component_ref 0x402a08c0 type <pointer_type 0x401eec40>
            arg 0 <indirect_ref 0x402aefc8 type <record_type 0x4026be00 _IO_FILE>
                arg 0 <parm_decl 0x402af930 __fp>>
            arg 1 <field_decl 0x40284930
                _IO_read_end type <pointer_type 0x401eec40>
                unsigned in_system_header SI
                file /usr/include/libio.h line 268
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 offset_align 128 offset <integer_cst 0x401ed5c0 0>
                bit offset <integer_cst 0x401ed2c0 constant 64>
                context <record_type 0x4026be00 _IO_FILE>
                arguments <integer_cst 0x401ed5c0 0>
                chain <field_decl 0x402849a0 _IO_read_base>>>>
    arg 1 <call_expr 0x402a0900 type <integer_type 0x401da380 int>
        side-effects
        arg 0 <addr_expr 0x402b0014
            type <pointer_type 0x402af8c0
                type <function_type 0x4028e230 type <integer_type 0x401da380 int>
                    DI size <integer_cst 0x401ed2c0 64>
                    unit size <integer_cst 0x401ed4e0 constant 8>
                    align 64 symtab 0 alias set -1
```

```
                    arg-types <tree_list 0x4028d190
                        value <pointer_type 0x4028e150
                            type <record_type 0x4028a690
                                _IO_FILE type_0 BLK
                                size <integer_cst 0x40288400 1184>
                                unit size <integer_cst 0x402883c0 148>
                                align 32 symtab 0 alias set -1
                                fields <field_decl 0x40284850 _flags>
                                pointer_to_this <pointer_type 0x4028e150>>
                            unsigned SI size <integer_cst 0x401ed540 32>
                            unit size <integer_cst 0x401ed5a0 4>
                            align 32 symtab 0 alias set -1>
                        chain <tree_list 0x4028d1a4
                            value <void_type 0x401ee770 void VOID
                                align 8 symtab 0 alias set -1
                                pointer_to_this <pointer_type 0x401ee7e0>>>>
                    pointer_to_this <pointer_type 0x402af8c0>>
                unsigned SI size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1>

            arg 0 <function_decl 0x4028e3f0 __uflow type <function_type 0x4028e230>
                used public in_system_header common external
                defer-output QI file /usr/include/libio.h line 397
                chain <function_decl 0x4028e2a0 __underflow>>>
        arg 1 <tree_list 0x402b003c
            value <nop_expr 0x402b0028 type <pointer_type 0x4028e150>
                arg 0 <parm_decl 0x402af930 __fp>>>>
    arg 2 <nop_expr 0x402b012c type <integer_type 0x401da380 int>
        side-effects
        arg 0 <indirect_ref 0x402b0118
            type <integer_type 0x401da1c0 unsigned char unsigned QI
            size <integer_cst 0x401eada0 8>
            unit size <integer_cst 0x401eadc0 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae20 0> max <integer_cst 0x401eae40 255>
                pointer_to_this <pointer_type 0x40271af0>>
            side-effects
            arg 0 <nop_expr 0x402b0104
                type <pointer_type 0x40271af0
                    type <integer_type 0x401da1c0 unsigned char>
                    unsigned SI size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 symtab 0 alias set -1
                    pointer_to_this <pointer_type 0x40271b60>>
                side-effects
                arg 0 <postincrement_expr 0x402a0960 type <pointer_type 0x401eec40>
                    side-effects
                    arg 0 <component_ref 0x402a0920 type <pointer_type 0x401eec40>

                        arg 0 <indirect_ref 0x402b00c8
                                type <record_type 0x4026be00 _IO_FILE>
```

```
                        arg 0 <parm_decl 0x402af930 __fp>>
                        arg 1 <field_decl 0x402848c0 _IO_read_ptr>>
                arg 1 <integer_cst 0x402a0940 constant 1>>>>>>
<call_expr 0x4030f340
   type <real_type 0x401ee9a0 double DF
       size <integer_cst 0x401ed2c0 constant 64>
       unit size <integer_cst 0x401ed4e0 constant 8>
       align 64 symtab 0 alias set -1 precision 64>
   side-effects
   arg 0 <addr_expr 0x403117a8
       type <pointer_type 0x40310930
           type <function_type 0x402fc930 type <real_type 0x401ee9a0 double>
               DI size <integer_cst 0x401ed2c0 64>
               unit size <integer_cst 0x401ed4e0 8>
               align 64 symtab 0 alias set -1
               arg-types <tree_list 0x402fd3fc
                   value <pointer_type 0x40290690
                       type <integer_type 0x401f1700 char readonly QI
                           size <integer_cst 0x401eada0 constant 8>
                           unit size <integer_cst 0x401eadc0 constant 1>
                           align 8 symtab 0 alias set -1 precision 8
                           min <integer_cst 0x401eae60 -128>
                           max <integer_cst 0x401eae80 127>
                           pointer_to_this <pointer_type 0x401f1770>>
                       unsigned SI
                       size <integer_cst 0x401ed540 constant 32>
                       unit size <integer_cst 0x401ed5a0 constant 4>
                       align 32 symtab 0 alias set -1>
                   chain <tree_list 0x402fd410
                       value <pointer_type 0x40297a10
                           type <pointer_type 0x401eec40
                               type <integer_type 0x401da230 char QI
                                   size <integer_cst 0x401eada0 8>
                                   unit size <integer_cst 0x401eadc0 1>
                                   align 8 symtab 0 alias set -1 precision 8
                                   min <integer_cst 0x401eae60 -128>
                                   max <integer_cst 0x401eae80 127>
                                   pointer_to_this <pointer_type 0x401eec40>>
                               unsigned SI size <integer_cst 0x401ed540 32>
                               unit size <integer_cst 0x401ed5a0 4>
                               align 32 symtab 0 alias set -1
                               pointer_to_this <pointer_type 0x402979a0>>
                           unsigned SI size <integer_cst 0x401ed540 32>
                           unit size <integer_cst 0x401ed5a0 4>
                           align 32 symtab 0 alias set -1>
                       chain <tree_list 0x402fd424
                           value <void_type 0x401ee770 void VOID
                               align 8 symtab 0 alias set -1
                               pointer_to_this <pointer_type 0x401ee7e0>>>>>
               pointer_to_this <pointer_type 0x40310930>>
           unsigned SI size <integer_cst 0x401ed540 32>
           unit size <integer_cst 0x401ed5a0 4>
```

```
            align 32 symtab 0 alias set -1>
        arg 0 <function_decl 0x402fc9a0 strtod type <function_type 0x402fc930>
            used public in_system_header external inline QI
            file /usr/include/stdlib.h line 295
            arguments <parm_decl 0x4030bd90 __nptr type <pointer_type 0x40290690>
                unsigned used in_system_header SI
                file /usr/include/stdlib.h line 294
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 alias set -2 context <function_decl 0x402fc9a0 strtod>
                result <pointer_type 0x40290690>
                initial <pointer_type 0x40290690>
                arg-type <pointer_type 0x40290690>
                arg-type-as-written <pointer_type 0x40290690>
                chain <parm_decl 0x4030be00 __endptr>>
            result <result_decl 0x4030bf50 type <real_type 0x401ee9a0 double>
                in_system_header DF file /usr/include/stdlib.h line 295
                size <integer_cst 0x401ed2c0 64>
                unit size <integer_cst 0x401ed4e0 8>
                align 64 context <function_decl 0x402fc9a0 strtod>>
                initial <block 0x402f9480>
            (mem:QI (symbol_ref:SI ("strtod")) [0 S1 A8])
                chain <function_decl 0x402fc7e0 atoll>>>
    arg 1 <tree_list 0x403117bc
        value <parm_decl 0x40310770 __nptr
            type <pointer_type 0x401f1770 type <integer_type 0x401f1700 char>
                unsigned SI size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x4027ee00>>
            unsigned used in_system_header SI
            file /usr/include/stdlib.h line 355 size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402fc3f0 atof>
            result <pointer_type 0x401f1770> initial <pointer_type 0x401f1770>
            arg-type <pointer_type 0x401f1770>
            arg-type-as-written <pointer_type 0x401f1770>>
        chain <tree_list 0x403117d0
            value <integer_cst 0x4030f320 constant 0>>>>
```

BIND_EXPR e 3

RTL 을 만들고 공간을 할당하는 것을 포함하여, 지역 변수를 선언한다.
Operand 0 은 변수들을 위한 VAR_DECL node 들의 chain 이다.
Operand 1 은 변수들을 사용하여 계산되어진 body 이고 표현식이다. operand 1 의 값은 BIND_EXPR 의 것으로 된다.
Operand 2 은 디버깅 목적을 위한 그러한 binding 들과 상응하는 BLOCK 이다. 만약 이 BIND_EXPR 가 실제로 확장된다면 그것은 BLOCK 내의 TREE_USED flag 를 설정한다.

BIND_EXPR 는 그러한 변수들에 대해 파서에 알릴 책임은 없다. 만약 body 가 입력 파일로부터 오고 있다면 BIND_EXPR 를 생성하는 code 는 또한 변수들의 파서를 알릴 책임이 있다.

만약 BIND_EXPR 가 일찍이 확장되었다면, 그것읜 TREE_USED flag 은 설정된다. 이것은 debugging symbol table 에 대한 code 에게 BIND_EXPR 를 무시하지 말라고 말한다.

만약 BIND_EXPR 가 debugging 을 위해 반드시 output 해야 하지만 확장되지는 않을 것이라면 TREE_USED flag 를 손수 설정하라.

BIND_EXPR 가 모두에게 알려지기 위해서는, 그것을 생성한 code 는 반드시 함수를 위한 BLOCK node 들의 tree 내의 subblock 으로써 그것을 역시 설치 해야한다.

C 에서 사용되지 않는 TREE node 입니다.

CALL_EXPR e 2

Function call. Operand 0 은 함수입니다. Operand 1 은 TREE_LIST node 들의 chain 으로 구성된 expression 들의 list 인 argument list 입니다.

```
<call_expr 0x402a07a0
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eaf60 -2147483648>
       max <integer_cst 0x401eaf80 2147483647>
       pointer_to_this <pointer_type 0x401f1620>>
   side-effects
   arg 0 <addr_expr 0x402aeb90
       type <pointer_type 0x402af690
           type <function_type 0x40299f50 type <integer_type 0x401da380 int>
               DI
               size <integer_cst 0x401ed2c0 constant 64>
               unit size <integer_cst 0x401ed4e0 constant 8>
               align 64 symtab 0 alias set -1
               attributes <tree_list 0x4029a4c4
                   purpose <identifier_node 0x401ecb80 format>
                   value <tree_list 0x401f4154
                       value <identifier_node 0x401eca80 printf public
                           global <function_decl 0x401f95b0 printf
                               type <function_type 0x401f9540
                                   type <integer_type 0x401da380 int>
                                   DI size <integer_cst 0x401ed2c0 64>
                                   unit size <integer_cst 0x401ed4e0 8>
                                   align 64 symtab 0 alias set -1
                                   attributes <tree_list 0x401f430c
                                       purpose <identifier_node 0x401ecb80 format>
                                       value <tree_list 0x401f412c
                                           value <identifier_node 0x401eca80
                                               printf>
                                           chain <tree_list 0x401f40a0
                                               value <integer_cst 0x401ede60
                                                   constant 1>
                                               chain <tree_list 0x401f4050
                                                   value <integer_cst 0x401ede80
                                                       constant 2>>>>>
                                   arg-types <tree_list 0x401d8fa0
                                       value <pointer_type 0x401f1770
                                           type <integer_type 0x401f1700
```

```
                                        char readonly QI
                                        size <integer_cst 0x401eada0
                                            constant 8>
                                        unit size <integer_cst 0x401eadc0
                                            constant 1>
                                        align 8 symtab 0 alias set -1
                                        precision 8
                                        min <integer_cst 0x401eae60 -128>
                                        max <integer_cst 0x401eae80 127>
                                        pointer_to_this <pointer_type
                                            0x401f1770>>
                                    unsigned SI
                                    size <integer_cst 0x401ed540
                                        constant 32>
                                    unit size <integer_cst 0x401ed5a0
                                        constant 4>
                                    align 32 symtab 0 alias set -1
                                    pointer_to_this <pointer_type
                                        0x4027ee00>>>>
                            public in_system_header common external built-in
                            defer-output QI file include/stdio.h line 285
                            built-in BUILT_IN_FRONTEND:BUILT_IN_PRINTF
                            (mem:QI (symbol_ref:SI ("printf")) [0 S1 A8])
                            chain <function_decl 0x401f94d0 __builtin_printf>>>
                    chain <tree_list 0x401f40b4
                        value <integer_cst 0x401ede80 2>
                        chain <tree_list 0x401f4028
                            value <integer_cst 0x401ede40 constant 0>>>>>
            arg-types <tree_list 0x4029a460
                value <pointer_type 0x40295d20
                    type <record_type 0x4026bf50 FILE BLK
                        size <integer_cst 0x40288400 constant 1184>
                        unit size <integer_cst 0x402883c0 constant 148>
                        align 32 symtab 0 alias set -1
                        fields <field_decl 0x40284850 _flags
                            type <integer_type 0x401da380 int>
                            in_system_header SI
                            file /usr/include/libio.h line 262
                            size <integer_cst 0x401eaf00 32>
                            unit size <integer_cst 0x401eafa0 4>
                            align 32 offset_align 128
                            offset <integer_cst 0x401ed5c0 constant 0>
                            bit offset <integer_cst 0x401ed680 constant 0>
                            context <record_type 0x4026be00 _IO_FILE>
                            arguments <integer_cst 0x401ed5c0 0>
                            chain <field_decl 0x402848c0 _IO_read_ptr>>
                        pointer_to_this <pointer_type 0x40292930>>
                    unsigned SI size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 symtab 0 alias set -1>
                chain <tree_list 0x4029a474
                    value <pointer_type 0x40290690
```

```
                            type <integer_type 0x401f1700 char>
                            unsigned SI size <integer_cst 0x401ed540 32>
                            unit size <integer_cst 0x401ed5a0 4>
                            align 32 symtab 0 alias set -1>
                        chain <tree_list 0x4029a488
                            value <pointer_type 0x402522a0 __gnuc_va_list
                                type <integer_type 0x401da230 char QI
                                    size <integer_cst 0x401eada0 8>
                                    unit size <integer_cst 0x401eadc0 1>
                                    align 8 symtab 0 alias set -1 precision 8
                                    min <integer_cst 0x401eae60 -128>
                                    max <integer_cst 0x401eae80 127>
                                    pointer_to_this <pointer_type 0x401eec40>>
                                unsigned SI size <integer_cst 0x401ed540 32>
                                unit size <integer_cst 0x401ed5a0 4>
                                align 32 symtab 0 alias set -1>
                            chain <tree_list 0x4029a49c
                                value <void_type 0x401ee770 void VOID
                                    align 8 symtab 0 alias set -1
                                    pointer_to_this <pointer_type 0x401ee7e0>>>>>>
                pointer_to_this <pointer_type 0x402af690>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>

        arg 0 <function_decl 0x40299ee0 vfprintf type <function_type 0x40299f50>
            used public in_system_header common external defer-output QI
            file include/stdio.h line 292
            chain <function_decl 0x40299c40 sprintf>>>
    arg 1 <tree_list 0x402aeba4
        value <var_decl 0x40292a10 stdout
            type <pointer_type 0x40292930 type <record_type 0x4026bf50 FILE>
                unsigned SI size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1>
            unsigned used public in_system_header common external defer-output SI
            file include/stdio.h line 143
            size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 chain <var_decl 0x402929a0 stdin>>
        chain <tree_list 0x402aebb8
            value <parm_decl 0x402af3f0 __fmt type <pointer_type 0x40290690>
                unsigned used in_system_header SI
                file /usr/include/bits/stdio.h line 34
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 alias set -2 context <function_decl 0x4029b150 vprintf>
                result <pointer_type 0x40290690> initial <pointer_type 0x40290690>
                arg-type <pointer_type 0x40290690>
                arg-type-as-written <pointer_type 0x40290690>
                chain <parm_decl 0x402af460 __arg>>
            chain <tree_list 0x402aebcc
```

```
                value <parm_decl 0x402af460 __arg
                    type <pointer_type 0x402522a0 __gnuc_va_list>
                    unsigned used in_system_header SI
                    file /usr/include/bits/stdio.h line 34
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x4029b150 vprintf>
                    result <pointer_type 0x402522a0 __gnuc_va_list>
                    initial <pointer_type 0x402522a0 __gnuc_va_list>
                    arg-type <pointer_type 0x402522a0 __gnuc_va_list>
                    arg-type-as-written <pointer_type 0x402522a0
                        __gnuc_va_list>>>>>>
<call_expr 0x402a0820
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eaf60 -2147483648>
       max <integer_cst 0x401eaf80 2147483647>
       pointer_to_this <pointer_type 0x401f1620>>
   side-effects
   arg 0 <addr_expr 0x402aed84
       type <pointer_type 0x402af8c0
           type <function_type 0x4028e230 type <integer_type 0x401da380 int>
               DI
               size <integer_cst 0x401ed2c0 constant 64>
               unit size <integer_cst 0x401ed4e0 constant 8>
               align 64 symtab 0 alias set -1
               arg-types <tree_list 0x4028d190
                   value <pointer_type 0x4028e150
                       type <record_type 0x4028a690 _IO_FILE type_0 BLK
                           size <integer_cst 0x40288400 constant 1184>
                           unit size <integer_cst 0x402883c0 constant 148>
                           align 32 symtab 0 alias set -1
                           fields <field_decl 0x40284850 _flags
                               type <integer_type 0x401da380 int>
                               in_system_header SI
                               file /usr/include/libio.h line 262
                               size <integer_cst 0x401eaf00 32>
                               unit size <integer_cst 0x401eafa0 4>
                               align 32 offset_align 128
                               offset <integer_cst 0x401ed5c0 constant 0>
                               bit offset <integer_cst 0x401ed680 constant 0>
                               context <record_type 0x4026be00 _IO_FILE>
                               arguments <integer_cst 0x401ed5c0 0>
                               chain <field_decl 0x402848c0 _IO_read_ptr>>
                           pointer_to_this <pointer_type 0x4028e150>>
                       unsigned SI
                       size <integer_cst 0x401ed540 constant 32>
                       unit size <integer_cst 0x401ed5a0 constant 4>
                       align 32 symtab 0 alias set -1>
                   chain <tree_list 0x4028d1a4
```

```
                        value <void_type 0x401ee770 void VOID
                            align 8 symtab 0 alias set -1
                            pointer_to_this <pointer_type 0x401ee7e0>>>>
                    pointer_to_this <pointer_type 0x402af8c0>>
                unsigned SI size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1>
            arg 0 <function_decl 0x4028eb60 _IO_getc type <function_type 0x4028e230>
                used public in_system_header common external defer-output QI
                file /usr/include/libio.h line 426
                chain <function_decl 0x4028ea10 __woverflow>>>
        arg 1 <tree_list 0x402aedac
            value <nop_expr 0x402aed98 type <pointer_type 0x4028e150>
                arg 0 <var_decl 0x402929a0 stdin
                    type <pointer_type 0x40292930
                        type <record_type 0x4026bf50 FILE BLK
                            size <integer_cst 0x40288400 1184>
                            unit size <integer_cst 0x402883c0 148>
                            align 32 symtab 0 alias set -1
                            fields <field_decl 0x40284850 _flags>
                            pointer_to_this <pointer_type 0x40292930>>
                        unsigned SI size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 symtab 0 alias set -1>
                    unsigned used public in_system_header common external
                    defer-output SI file include/stdio.h line 142
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 chain <type_decl 0x40292850 fpos64_t>>>>>
```

METHOD_CALL_EXPR e 4

어떤 Method 를 호출한다.
Operand 0 은 type 이 METHOD_TYPE 인 method 이다.
Operand 1 는 "자신" 을 위한 표현식이다.
Operand 2 는 분명한 (explicit) argument 들의 list 이다.

C 에서 사용되지 않는 TREE node 입니다.

WITH_CLEANUP_EXPR e 3

상응하는 cleanup 에 따라 계산하는 값을 지정한다.
Operand 0 argument 는 값이 cleanup 이 필요한 것의 표현식이다.
Operand 1 은 object 을 위한 cleanup expression 이다.
Operand 2 은 결과적으로 해당 값을 나타낼 RTL_EXPR 이다.

> RTL_EXPR 는 이 표현식에서 사용되며, 이것은 어떻게 표현식이 적당한 값으로 행동하는지를 다루는 지에 대해서이다.

cleanup 는 CLEANUP_POINT_EXPR 가 존재할 경우 첫음 동본된 것으로 실행이 되며, 그렇지 않을 경우, 필요에 한해 그것은 수동으로 expand_start_target_temps/expand_end_target_temps 들을 호출하는 것은 caller 의 책임이다.

이것은 cleanup 들이 실행하여 exception 이 던져지지 않을 경우 operand 2 가 항상 평가되어져서 TRY_CATCH_EXPR 와는 다르다.

이 TREE node 를 위한 적당한 예제를 찾지 못하였습니다. 적당한 예제를 찾으신 분은 저에게 메일로 보내주시기 바랍니다.

CLEANUP_POINT_EXPR e 1

Cleanup point 를 지정한다.
Operand 0 은 아마 cleanup 들을 가지는 표현식이다. 만약 그럴경우, 그러한 cleanup 들은 표현식이 확장된 후 실행이 될것이다.

만약 표현식이 storage 에 대한 reference 라면, cleanup 들이 실행되기 전에 강제로 메모리가 고갈될 것임을 알고 있어라. 이것은 cleanup 들이 참조되어 지는 storage 를 수정할 수 있는 경우들을 다루기 위해 필요한다; 표현식 't.i' 에서 만약 't' 가 정수 멤버 'i' 를 가지는 구조체이고 cleanup 이 'i' 를 수정한다면, 표현식의 값은 cleanup 이 't.i' 가 평가되기 전에 실행되었는지 뒤에 실행되었는지에 따라 의존한다. expand_expr 가 't.i' 를 실행 중일 경우 그것은 MEM 을 반환한다. 이것은 충분히 좋지 못하다; 't.i' 의 값은 반드시 강제로 메모리 고갈되어야 한다.

결과적으로, CLEANUP_POINT_EXPR 의 operand 는 절대 BLKmode 를 가져서는 안되는데, 그것은 강제로 메모리 고갈이 되지 않기 때문이다.

이 TREE node 를 위한 적당한 예제를 찾지 못하였습니다. 적당한 예제를 찾으신 분은 저에게 메일로 보내주시기 바랍니다.

다음의 두 code 는 다른 field 의 offset 혹은 size 의 계산 과/혹은 type 의 size 에 사용되는 값을 표현하는 type 의 object 내의 몇몇 field 상에서 type 들을 가지는 언어들한테서 사용된다. Field 들의 위치들 과/혹은 size 들은 같은 type 의 object 부터 object 까지 여러가지가 될 수 있다.
Ada 의 판별식이나 Pascal 내의 schema type 들을 가진 record type 들은 그러한 type 들의 예이다. 이 메카니즘은 또한 Ada 내의 구속받지 않은 배열 type 들에 대한 "fat pointer 들" 을 생성하는데 사용된다; fat pointer 는 구조체인데, 그러한 field 들의 하나가 실제 배열 type 으로의 포인터이고 다른 field 은 배열의 경계를 포함하는 구조체 형틀로의 포인터 이다. fat pointer 내의 처음 field 에 의해 가르켜지는 type 내의 경계는 형틀내의 값을 참조한다.
당신이 그러한 type 을 건설하고자 희망할 경우 당신은 TYPE node 로 부터 이 type 을 가지는 object 를 참조하게 허락하는 "자기-참조들" 이 필요하다. 예를 들면 이 type 의 실증을 드는 변수를 가지지 않는 것 말이다.
그러한 "자기-참조들" 은 PLACEHOLDER_EXPR 를 사용하여 되어진다. 이것은 나중에 참조되어지는 object 로 대체될 node 이다. 그것의 type 은 object 의 그것이고 reference 들의 chain 으로부터 어떤 object 를 사용할지를 선택한다. (밑을 봐라). 다른 slot 들은 PLACEHOLDER_EXPR 에서 사용되지 않는다.
예를 들어, 만약 당신의 type FOO 가 field BAR 를 가지는 RECORD_TYPE 이고 TYPE_SIZE (FOO) 를 계산하기 위한 ¡variable¿.BAR 의 값이 필요하다면, 단지 위의 ¡variable¿ 을 우리가 평가하기를 원하는 표현식과 object 가 검색된 것 중에서의 한 표현식을 포함하는 PLACEHOLDER_EXPR 로 대체하라. 후자 표현식은 판별식으로 기록하는 Ada 의 간단한 경우에는 그 자신의 object 이지만 구속박지 않는 배열의 경우일 때는 배열일 수 있다.
후자의 경우, 우리는 배열의 경계가 fat pointer 로부터만 접근되어질 수 있기 때문에 fat pointer 가 필요하다. 하지만 여기에서는 우리는 배열을 위한 표현식이 배열 포인터를 가지고 있는 fat pointer 의 dereference 를 포함하고 있다는 사실에 의존한다.
따라서, PLACEHOLDER_EXPR 의 자리를 대체할 object 를 찾을 때는, 우리는 원하는 type 의 어떤 것을 찾았거나 상수에 도달할때까지 WITH_RECORD_EXPR 에 두번째 operand 로 전해지는 표현식의 첫번째 operand 를 살펴보면 된다.

PLACEHOLDER_EXPR x 0

이 표현식을 평가할 때 나중에 WITH_RECORD_EXPR 에 의해 제공되어질 recode 를 표시한다. 이 표현식의 type 은 그것을 대체할 record 를 찾는데 사용된다.

C 에서 사용되지 않는 TREE node 입니다.

WITH_RECORD_EXPR e 2

PLACEHOLDER_EXPR 내에서 사용되는 record 를 참조하는 표현식을 제공한다. 사용되는 record 는 operand 0 내 PLACEHOLDER_EXPR 와 같은 type 을 가지는 operand 1 내부의 record 이다.

이 TREE node 를 위한 적당한 예제를 찾지 못하였습니다. 적당한 예제를 찾으신 분은 저에게 메일로 보내주시기 바랍니다.

PLUS_EXPR 2 2

간단한 산술연산. 더하기

```
<plus_expr 0x40373260
   type <pointer_type 0x401ee7e0
       type <void_type 0x401ee770 void VOID
           align 8 symtab 0 alias set -1
           pointer_to_this <pointer_type 0x401ee7e0>>
       unsigned SI
       size <integer_cst 0x401ed540 constant 32>
       unit size <integer_cst 0x401ed5a0 constant 4>
       align 32 symtab 0 alias set -1
       pointer_to_this <pointer_type 0x402801c0>>
   arg 0 <minus_expr 0x40373220 type <pointer_type 0x401ee7e0>
       arg 0 <parm_decl 0x403760e0 f
           type <pointer_type 0x4036e230
               type <record_type 0x4036e150 dwarf_fde packed type_0 BLK
                   size <integer_cst 0x401ed2c0 constant 64>
                   unit size <integer_cst 0x401ed4e0 constant 8>
                   user align 32 symtab 0 alias set -1
                   attributes <tree_list 0x4037503c
                       purpose <identifier_node 0x40374200 aligned>
                       value <tree_list 0x40375000
                           value <integer_cst 0x403730c0 constant 4>>
                       chain <tree_list 0x40375028
                           purpose <identifier_node 0x402d0fc0 packed>>>
                   fields <field_decl 0x40372d90 length
                       type <integer_type 0x40372700 uword unsigned SI
                           size <integer_cst 0x401eaf00 constant 32>
                           unit size <integer_cst 0x401eafa0 constant 4>
                           align 32 symtab 0 alias set -1 precision 32
                           min <integer_cst 0x401eafe0 0>
                           max <integer_cst 0x401ed000 4294967295>>
                       unsigned packed SI file unwind-dw2-fde.h line 141
                       size <integer_cst 0x401eaf00 32>
                       unit size <integer_cst 0x401eafa0 4>
                       align 8 offset_align 128
                       offset <integer_cst 0x401ed5c0 constant 0>
                       bit offset <integer_cst 0x401ed680 constant 0>
                       context <record_type 0x4036e150 dwarf_fde>
                       arguments <integer_cst 0x401ed5c0 0>
                       chain <field_decl 0x40372e00 CIE_delta>>
                   pointer_to_this <pointer_type 0x4036e230>
```

```
                    chain <type_decl 0x4036e1c0>>
                unsigned SI size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x4036e310>>
            unsigned used SI file unwind-dw2-fde.h line 151
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x40376230 get_cie>
            result <pointer_type 0x4036e230> initial <pointer_type 0x4036e230>
            arg-type <pointer_type 0x4036e230>
            arg-type-as-written <pointer_type 0x4036e230>>
        arg 1 <component_ref 0x40373180
            type <integer_type 0x40372620 sword SI size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401eaf60 -2147483648>
                max <integer_cst 0x401eaf80 2147483647>
                pointer_to_this <pointer_type 0x40376310>>
            arg 0 <indirect_ref 0x40375398 type <record_type 0x4036e150 dwarf_fde>
                arg 0 <parm_decl 0x403760e0 f>>
            arg 1 <field_decl 0x40372e00 CIE_delta
                type <integer_type 0x40372620 sword>
                packed SI file unwind-dw2-fde.h line 142
                size <integer_cst 0x401eaf00 32>
                unit size <integer_cst 0x401eafa0 4>
                align 8 offset_align 128 offset <integer_cst 0x401ed5c0 0>
                bit offset <integer_cst 0x401eaf00 32>
                context <record_type 0x4036e150 dwarf_fde>
                arguments <integer_cst 0x401ed5c0 0>
                chain <field_decl 0x40372f50 pc_begin>>>>
    arg 1 <integer_cst 0x40373240 type <pointer_type 0x401ee7e0> constant 4>>
<plus_expr 0x40373420
    type <pointer_type 0x401eec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128>
            max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402979a0>>
    arg 0 <plus_expr 0x40373340 type <pointer_type 0x401eec40>
        arg 0 <nop_expr 0x403756e0 type <pointer_type 0x401eec40>
            arg 0 <parm_decl 0x403763f0 f
                type <pointer_type 0x40376380
                    type <record_type 0x40376070 fde packed type_0 BLK
                        size <integer_cst 0x401ed2c0 constant 64>
                        unit size <integer_cst 0x401ed4e0 constant 8>
```

```
                        user align 32 symtab 0 alias set -1
                        attributes <tree_list 0x4037503c
                            purpose <identifier_node 0x40374280 aligned>
                            value <tree_list 0x40375000
                                value <integer_cst 0x403730c0 constant 4>>
                            chain <tree_list 0x40375028
                                purpose <identifier_node 0x402d4040 packed>>>
                        fields <field_decl 0x40372d90 length
                            type <integer_type 0x40372700 uword unsigned SI
                                size <integer_cst 0x401eaf00 constant 32>
                                unit size <integer_cst 0x401eafa0 constant 4>
                                align 32 symtab 0 alias set -1 precision 32
                                min <integer_cst 0x401eafe0 0>
                                max <integer_cst 0x401ed000 4294967295>>
                            unsigned packed SI file unwind-dw2-fde.h line 141
                            size <integer_cst 0x401eaf00 32>
                            unit size <integer_cst 0x401eafa0 4>
                            align 8 offset_align 128
                            offset <integer_cst 0x401ed5c0 constant 0>
                            bit offset <integer_cst 0x401ed680 constant 0>
                            context <record_type 0x4036e150 dwarf_fde>
                            arguments <integer_cst 0x401ed5c0 0>
                            chain <field_decl 0x40372e00 CIE_delta>>
                        pointer_to_this <pointer_type 0x40376380>>
                    unsigned SI size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 symtab 0 alias set -1>
                unsigned used SI file unwind-dw2-fde.h line 157
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x403764d0 next_fde>
                result <pointer_type 0x40376380> initial <pointer_type 0x40376380>
                arg-type <pointer_type 0x40376380>
                arg-type-as-written <pointer_type 0x40376380>>>
            arg 1 <convert_expr 0x40375744 type <pointer_type 0x401eec40>
                arg 0 <non_lvalue_expr 0x40375730 type <integer_type 0x40372700 uword>
                    arg 0 <component_ref 0x403732e0
                        type <integer_type 0x40372700 uword>
                        arg 0 <indirect_ref 0x403756f4
                            type <record_type 0x4036e150 dwarf_fde packed type_0 BLK
                                size <integer_cst 0x401ed2c0 64>
                                unit size <integer_cst 0x401ed4e0 8>
                                user align 32 symtab 0 alias set -1
                                attributes <tree_list 0x4037503c>
                                fields <field_decl 0x40372d90 length>
                                pointer_to_this <pointer_type 0x4036e230>
                                chain <type_decl 0x4036e1c0>>
                            arg 0 <parm_decl 0x403763f0 f>>
                        arg 1 <field_decl 0x40372d90 length>>>>>
        arg 1 <integer_cst 0x40373400 type <pointer_type 0x401eec40> constant 4>>
```

MINUS_EXPR 2 2

간단한 산술연산. 빼기

```
<minus_expr 0x401d7ee0
   type <integer_type 0x401de4d0 unsigned int unsigned sizetype SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d7620 0> max <integer_cst 0x401d7640 4294967295>>
   constant
   arg 0 <integer_cst 0x401d7d80
       type <integer_type 0x401de4d0 unsigned int> constant 3>
   arg 1 <integer_cst 0x401d7c00
       type <integer_type 0x401de4d0 unsigned int> constant 0>>
<minus_expr 0x4031fa40
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eaf60 -2147483648>
       max <integer_cst 0x401eaf80 2147483647>
       pointer_to_this <pointer_type 0x401f1620>>
   constant
   arg 0 <integer_cst 0x4031f9e0 type <integer_type 0x401da380 int> constant 3>
   arg 1 <integer_cst 0x4031fa20 type <integer_type 0x401da380 int> constant 1>>
<minus_expr 0x401e05c0
   type <real_type 0x401de930 float SF
       size <integer_cst 0x401d7b80 constant 32>
       unit size <integer_cst 0x401d7be0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       pointer_to_this <pointer_type 0x401dfcb0>>
   arg 0 <var_decl 0x401f6540 i type <real_type 0x401de930 float>
       asm_written used public static common SF file <stdin> line 1
       size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
       align 32 initial <real_cst 0x401f5a40 1.39999997615814208984e0>
       (mem/f:SF (symbol_ref:SI ("i")) [0 i+0 S4 A32])
       chain <type_decl 0x401eed90 __g77_ulongint
           type <integer_type 0x401da5b0 long long unsigned int unsigned DI
               size <integer_cst 0x401d7600 constant 64>
               unit size <integer_cst 0x401d7720 constant 8>
               align 64 symtab 0 alias set -1 precision 64
               min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
           VOID file <built-in> line 0
           align 1 chain <type_decl 0x401eed20 __g77_longint>>>
   arg 1 <var_decl 0x401f65b0 j type <real_type 0x401de930 float>
       asm_written used public static common SF file <stdin> line 1
       size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
       align 32 initial <real_cst 0x401f5b00 9.00000000000000000000e0>
       (mem/f:SF (symbol_ref:SI ("j")) [0 j+0 S4 A32])
       chain <var_decl 0x401f6540 i>>>
```

MULT_EXPR 2 2

간단한 산술연산. 곱하기

```
<mult_expr 0x402c3320
   type <integer_type 0x401da3f0 unsigned int unsigned SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
       pointer_to_this <pointer_type 0x402c17e0>>
   constant
   arg 0 <integer_cst 0x402c3300
       type <integer_type 0x401da3f0 unsigned int> constant 8>
   arg 1 <integer_cst 0x402c32e0
       type <integer_type 0x401da3f0 unsigned int> constant 4>>
<mult_expr 0x402e50c0
   type <integer_type 0x402620e0 size_t unsigned SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
       pointer_to_this <pointer_type 0x40281c40>>

   arg 0 <var_decl 0x402e38c0 __result type <integer_type 0x402620e0 size_t>
       unsigned used in_system_header common regdecl SI
       file /usr/include/bits/string2.h line 921
       size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
       align 32 context <function_decl 0x402e3620 __strcspn_c1>
       initial <integer_cst 0x402e5060 0>>
   arg 1 <integer_cst 0x402e50a0
       type <integer_type 0x402620e0 size_t> constant 1>>
<mult_expr 0x401e05c0
   type <real_type 0x401de930 float SF
       size <integer_cst 0x401d7b80 constant 32>
       unit size <integer_cst 0x401d7be0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       pointer_to_this <pointer_type 0x401dfcb0>>
   arg 0 <var_decl 0x401f6540 i type <real_type 0x401de930 float>
       asm_written used public static common SF file <stdin> line 1
       size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
       align 32 initial <real_cst 0x401f5a40 1.399999976158142089 84e0>
       (mem/f:SF (symbol_ref:SI ("i")) [0 i+0 S4 A32])
       chain <type_decl 0x401eed90 __g77_ulongint
           type <integer_type 0x401da5b0 long long unsigned int unsigned DI
               size <integer_cst 0x401d7600 constant 64>
               unit size <integer_cst 0x401d7720 constant 8>
               align 64 symtab 0 alias set -1 precision 64
               min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
           VOID file <built-in> line 0
           align 1 chain <type_decl 0x401eed20 __g77_longint>>>
   arg 1 <var_decl 0x401f65b0 j type <real_type 0x401de930 float>
       asm_written used public static common SF file <stdin> line 1
       size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
       align 32 initial <real_cst 0x401f5b00 9.0000000000000000000e0>
       (mem/f:SF (symbol_ref:SI ("j")) [0 j+0 S4 A32])
```

```
        chain <var_decl 0x401f6540 i>>>
```

TRUNC_DIV_EXPR 2 2

0 으로의 몫을 반올림한 정수 결과에 대한 나누기.

```
 <trunc_div_expr 0x402c35c0
    type <integer_type 0x401da3f0 unsigned int unsigned SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
        pointer_to_this <pointer_type 0x402c17e0>>
    constant
    arg 0 <integer_cst 0x402c35a0
        type <integer_type 0x401da3f0 unsigned int> constant 1024>
    arg 1 <integer_cst 0x402c3580
        type <integer_type 0x401da3f0 unsigned int> constant 32>>
 <trunc_div_expr 0x401e0600
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    arg 0 <var_decl 0x401f6540 i type <integer_type 0x401da380 int>
        asm_written used public static common SI file <stdin> line 1
        size <integer_cst 0x401d7540 32> unit size <integer_cst 0x401d75e0 4>
        align 32 initial <integer_cst 0x401e05c0 1>
        (mem/f:SI (symbol_ref:SI ("i")) [0 i+0 S4 A32])
        chain <type_decl 0x401eed90 __g77_ulongint
            type <integer_type 0x401da5b0 long long unsigned int unsigned DI
                size <integer_cst 0x401d7600 constant 64>
                unit size <integer_cst 0x401d7720 constant 8>
                align 64 symtab 0 alias set -1 precision 64
                min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
            VOID file <built-in> line 0
            align 1 chain <type_decl 0x401eed20 __g77_longint>>>
    arg 1 <var_decl 0x401f65b0 j type <integer_type 0x401da380 int>
        asm_written used public static common SI file <stdin> line 1
        size <integer_cst 0x401d7540 32> unit size <integer_cst 0x401d75e0 4>
        align 32 initial <integer_cst 0x401e05e0 9>
        (mem/f:SI (symbol_ref:SI ("j")) [0 j+0 S4 A32])
        chain <var_decl 0x401f6540 i>>>
```

CEIL_DIV_EXPR 2 2

무한대로의 몫을 반올림한 정수 결과에 대한 나누기.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적당한 예제를 가지고 계신 분은 저에게 메일로 보내주시면 감사하겠습니다.

FLOOR_DIV_EXPR 2 2

Minus 무한대로의 몫을 반올림한 정수 결과에 대한 나누기.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적당한 예제를 가지고 계신 분은 저에게 메일로 보내주시면 감사하겠습니다.

ROUND_DIV_EXPR 2 2

근처 정수로의 몫을 반올림한 정수 결과에 대한 나누기.

C 에서 사용되지 않는 TREE node 입니다.

TRUNC_MOD_EXPR 2 2

0 으로의 몫을 반올림한 정수 결과에 대한 나머지.

```
<trunc_mod_expr 0x401e05e0
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   constant
   arg 0 <integer_cst 0x401d7c80 type <integer_type 0x401da380 int> constant 1>
   arg 1 <integer_cst 0x401e05c0 type <integer_type 0x401da380 int> constant 9>>
<trunc_mod_expr 0x401e05e0
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   arg 0 <var_decl 0x401f6540 i type <integer_type 0x401da380 int>
       asm_written used public static common SI file <stdin> line 2
       size <integer_cst 0x401d7540 32> unit size <integer_cst 0x401d75e0 4>
       align 32 initial <integer_cst 0x401d7c80 1>
       (mem/f:SI (symbol_ref:SI ("i")) [0 i+0 S4 A32])
       chain <type_decl 0x401eed90 __g77_ulongint
           type <integer_type 0x401da5b0 long long unsigned int unsigned DI
               size <integer_cst 0x401d7600 constant 64>
               unit size <integer_cst 0x401d7720 constant 8>
               align 64 symtab 0 alias set -1 precision 64
               min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
           VOID file <built-in> line 0
           align 1 chain <type_decl 0x401eed20 __g77_longint>>>
   arg 1 <var_decl 0x401f65b0 j type <integer_type 0x401da380 int>
       asm_written used public static common SI file <stdin> line 2
       size <integer_cst 0x401d7540 32> unit size <integer_cst 0x401d75e0 4>
       align 32 initial <integer_cst 0x401e05c0 9>
       (mem/f:SI (symbol_ref:SI ("j")) [0 j+0 S4 A32])
       chain <var_decl 0x401f6540 i>>>
```

CEIL_MOD_EXPR 2 2

무한대로의 몫을 반올림한 정수 결과에 대한 나머지.

C 에서 사용되지 않는 TREE node 들 입니다.

FLOOR_MOD_EXPR 2 2

Minus 무한대로의 몫을 반올림한 정수 결과에 대한 나머지.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적당한 예제를 가지고 계신 분은 저에게 메일로 보내주시면 감사하겠습니다.

ROUND_MOD_EXPR 2 2

근처 정수로의 몫을 반올림한 정수 결과에 대한 나머지.

C 에서 사용되지 않는 TREE node 들 입니다.

RDIV_EXPR 2 2

실수 결과에 대한 나누기.

```
<rdiv_expr 0x401e05c0
   type <real_type 0x401de9a0 double DF
       size <integer_cst 0x401d7900 constant 64>
       unit size <integer_cst 0x401d7b20 constant 8>
       align 64 symtab 0 alias set -1 precision 64>
   constant
   arg 0 <real_cst 0x401f5a00 type <real_type 0x401de9a0 double>
       constant 9.4000000000000035527e0>
   arg 1 <real_cst 0x401f5a40 type <real_type 0x401de9a0 double>
       constant 2.6000000000000008882e0>>
<rdiv_expr 0x401e05c0
   type <real_type 0x401de930 float SF
       size <integer_cst 0x401d7b80 constant 32>
       unit size <integer_cst 0x401d7be0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       pointer_to_this <pointer_type 0x401dfcb0>>
   arg 0 <var_decl 0x401f6540 i type <real_type 0x401de930 float>
       asm_written used public static common SF file <stdin> line 1
       size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
       align 32 initial <real_cst 0x401f5a40 1.3999999761581420898e0>
       (mem/f:SF (symbol_ref:SI ("i")) [0 i+0 S4 A32])
       chain <type_decl 0x401eed90 __g77_ulongint
           type <integer_type 0x401da5b0 long long unsigned int unsigned DI
               size <integer_cst 0x401d7600 constant 64>
               unit size <integer_cst 0x401d7720 constant 8>
               align 64 symtab 0 alias set -1 precision 64
       min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
           VOID file <built-in> line 0
           align 1 chain <type_decl 0x401eed20 __g77_longint>>>
   arg 1 <var_decl 0x401f65b0 j type <real_type 0x401de930 float>
```

```
asm_written used public static common SF file <stdin> line 1
size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
align 32 initial <real_cst 0x401f5b00 9.0000000000000000000e0>
(mem/f:SF (symbol_ref:SI ("j")) [0 j+0 S4 A32])
chain <var_decl 0x401f6540 i>>>
```

EXACT_DIV_EXPR 2 2

반올림이 필요없는 것으로 간주되는 나누기. C 에서 포인터에 대한 빼기에 사용된다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

FIX_TRUNC_EXPR 1 1

실수를 고정점 (fixed point) 으로의 변환: 나누는 네가지 방법과 같이 반올림하는 네가지 방법. CONVERT_EXPR 또한 실수에서 정수로 변환하는데 사용될 수 있고, 그것이 원하는 것을 나타내는 방법을 가지고 있지 않은 언어들에서 사용 되는 것이다. 아마도 아래의 것은 필요 없을 수 있다.

```
<fix_trunc_expr 0x401e54ec
   type <integer_type 0x401da460 long int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d7660 -2147483648>
       max <integer_cst 0x401d7680 2147483647>
       pointer_to_this <pointer_type 0x401e27e0>>
   constant
   arg 0 <real_cst 0x401f5b80
       type <real_type 0x401de9a0 double DF
           size <integer_cst 0x401d7900 constant 64>
           unit size <integer_cst 0x401d7b20 constant 8>
           align 64 symtab 0 alias set -1 precision 64>
       constant 2.3399999999999985789e1>>
```

FIX_CEIL_EXPR 1 1
FIX_FLOOR_EXPR 1 1
FIX_ROUND_EXPR 1 1

실수를 고정점 (fixed point) 으로의 변환: 나누는 네가지 방법과 같이 반올림하는 네가지 방법. CONVERT_EXPR 또한 실수에서 정수로 변환하는데 사용될 수 있고, 그것이 원하는 것을 나타내는 방법을 가지고 있지 않은 언어들에서 사용 되는 것이다. 아마도 아래의 것은 필요 없을 수 있다.

C 에서 사용되지 않는 TREE node 들 입니다.

FLOAT_EXPR 1 1

정수에서 실수로의 변환.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다.

NEGATE_EXPR 1 1

Unary negation.

```
<negate_expr 0x40385424 type <integer_type 0x40382620 sword>
    arg 0 <component_ref 0x40383180 type <integer_type 0x40382620 sword>
        arg 0 <indirect_ref 0x40385398 type <record_type 0x4037e150 dwarf_fde>
            arg 0 <parm_decl 0x403860e0 f>>
        arg 1 <field_decl 0x40382e00 CIE_delta
            type <integer_type 0x40382620 sword>
            packed SI file unwind-dw2-fde.h line 142
            size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
            align 8 offset_align 128 offset <integer_cst 0x401ed5c0 0>
            bit offset <integer_cst 0x401eaf00 32>
            context <record_type 0x4037e150 dwarf_fde>
            arguments <integer_cst 0x401ed5c0 0>
            chain <field_decl 0x40382f50 pc_begin>>>>
```

MIN_EXPR 2 2
MAX_EXPR 2 2

아직 설명이 없음.

이 TREE node 의 경우, C++ 상에서 사용되며, C 에서는 사용되지 않습니다.

ABS_EXPR 1 1

Operand 의 절대값을 나타낸다.

ABS_EXPR 는 반드시 INTEGER_TYPE 혹은 REAL_TYPE 중 하나를 가져야 한다.
ABS_EXPR 의 operand 는 반드시 같은 type 을 가져야 한다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 찾으시는 분은 저에게 메일 주시면 감사하겠습니다.

FFS_EXPR 1 1

아직 설명이 없음.

C 에서 사용되지 않는 TREE node 입니다.

LSHIFT_EXPR 2 2

Shift 와 rotate 를 위한 shift operation 들. Shift 는 unsigned type 상에서 이루어진다면 logical shift 이고, signed type 상에서 이루어진다면 arithmetic shift 이다. 두번째 operand 는 shift 할 bit 들의 갯수이다; 이것은 첫번째 operand 와 결과와 같은 type 일 필요는 없다. 만약 두번째 operand 가 첫번째 operand 의 type size 보다 클 경우 결과는 정의되지 않음을 알기 바란다.

```
 <lshift_expr 0x401e0660
    type <integer_type 0x401da150 signed char QI
        size <integer_cst 0x401d73e0 constant 8>
        unit size <integer_cst 0x401d7400 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401d73a0 -128> max <integer_cst 0x401d73c0 127>>
    arg 0 <nop_expr 0x401e54c4 type <integer_type 0x401da150 signed char>
        arg 0 <var_decl 0x401f6540 ch
```

```
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401d73e0 8>
                unit size <integer_cst 0x401d7400 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401d74a0 -128>
                max <integer_cst 0x401d74c0 127>
                pointer_to_this <pointer_type 0x401dec40>>
            asm_written used public static common QI file <stdin> line 1
            size <integer_cst 0x401d73e0 8>
            unit size <integer_cst 0x401d7400 1>
            align 8 initial <integer_cst 0x401e05e0 -89>
            (mem/f:QI (symbol_ref:SI ("ch")) [0 ch+0 S1 A8])
            chain <type_decl 0x401eed90 __g77_ulongint>>>
        arg 1 <integer_cst 0x401e0640
            type <integer_type 0x401da150 signed char> constant 4>>
```

RSHIFT_EXPR 2 2

Shift 와 rotate 를 위한 shift operation 들. Shift 는 unsigned type 상에서 이루어진다면 logical shift 이고, signed type 상에서 이루어진다면 arithmetic shift 이다. 두번째 operand 는 shift 할 bit 들의 갯수이다; 이것은 첫번째 operand 와 결과와 같은 type 일 필요는 없다. 만약 두번째 operand 가 첫번째 operand 의 type size 보다 클 경우 결과는 정의되지 않음을 알기 바란다.

```
 <rshift_expr 0x401e0620
    type <integer_type 0x401da230 char QI
        size <integer_cst 0x401d73e0 constant 8>
        unit size <integer_cst 0x401d7400 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401d74a0 -128> max <integer_cst 0x401d74c0 127>
        pointer_to_this <pointer_type 0x401dec40>>
    arg 0 <var_decl 0x401f6540 ch type <integer_type 0x401da230 char>
        asm_written used public static common QI file <stdin> line 1
        size <integer_cst 0x401d73e0 8> unit size <integer_cst 0x401d7400 1>
        align 8 initial <integer_cst 0x401e05e0 -89>
        (mem/f:QI (symbol_ref:SI ("ch")) [0 ch+0 S1 A8])
        chain <type_decl 0x401eed90 __g77_ulongint
            type <integer_type 0x401da5b0 long long unsigned int unsigned DI
                size <integer_cst 0x401d7600 constant 64>
                unit size <integer_cst 0x401d7720 constant 8>
                align 64 symtab 0 alias set -1 precision 64
                min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>>
            VOID file <built-in> line 0
            align 1 chain <type_decl 0x401eed20 __g77_longint>>>
    arg 1 <integer_cst 0x401e0600
        type <integer_type 0x401da380 int> constant 2>>
```

LROTATE_EXPR 2 2
RROTATE_EXPR 2 2

Shift 와 rotate 를 위한 shift operation 들. Shift 는 unsigned type 상에서 이루어진다면 logical shift 이고, signed type 상에서 이루어진다면 arithmetic shift 이다. 두번째 operand 는 shift 할 bit 들의 갯수이다; 이것은 첫번째 operand 와 결과와 같은 type 일 필요는 없다. 만약 두번째 operand 가 첫번째 operand 의 type size 보다 클 경우 결과는 정의되지 않음을 알기 바란다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

BIT_IOR_EXPR 2 2

bit 관련 연산들. Operand 들은 결과로써 같은 mode 를 가진다.

```
<bit_ior_expr 0x40240540
   type <integer_type 0x401da230 char QI
       size <integer_cst 0x401d73e0 constant 8>
       unit size <integer_cst 0x401d7400 constant 1>
       align 8 symtab 0 alias set -1 precision 8
       min <integer_cst 0x401d74a0 -128> max <integer_cst 0x401d74c0 127>
       pointer_to_this <pointer_type 0x401dec40>>
   arg 0 <var_decl 0x40246540 ch type <integer_type 0x401da230 char>
       used common QI file <stdin> line 8
       size <integer_cst 0x401d73e0 8> unit size <integer_cst 0x401d7400 1>
       align 8 context <function_decl 0x402463f0 main>
       initial <integer_cst 0x40240400 65>>
   arg 1 <integer_cst 0x40240520
       type <integer_type 0x401da230 char> constant 32>>
```

BIT_XOR_EXPR 2 2

bit 관련 연산들. Operand 들은 결과로써 같은 mode 를 가진다.

```
<bit_xor_expr 0x401e0620
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   arg 0 <nop_expr 0x401e549c type <integer_type 0x401da380 int>
       arg 0 <var_decl 0x401f6540 ch1
           type <integer_type 0x401da230 char QI
               size <integer_cst 0x401d73e0 constant 8>
               unit size <integer_cst 0x401d7400 constant 1>
               align 8 symtab 0 alias set -1 precision 8
               min <integer_cst 0x401d74a0 -128>
               max <integer_cst 0x401d74c0 127>
               pointer_to_this <pointer_type 0x401dec40>>
           asm_written used public static common QI file <stdin> line 1
           size <integer_cst 0x401d73e0 8>
           unit size <integer_cst 0x401d7400 1>
           align 8 initial <integer_cst 0x401e05e0 60>
           (mem/f:QI (symbol_ref:SI ("ch1")) [0 ch1+0 S1 A8])
           chain <type_decl 0x401eed90 __g77_ulongint>>>
   arg 1 <var_decl 0x401f65b0 ch2 type <integer_type 0x401da380 int>
       asm_written used public static common SI file <stdin> line 3
       size <integer_cst 0x401d7540 32> unit size <integer_cst 0x401d75e0 4>
       align 32 initial <integer_cst 0x401e0600 43>
```

```
        (mem/f:SI (symbol_ref:SI ("ch2")) [0 ch2+0 S4 A32])
        chain <var_decl 0x401f6540 ch1>>>
```

BIT_AND_EXPR 2 2

bit 관련 연산들. Operand 들은 결과로써 같은 mode 를 가진다.

```
 <bit_and_expr 0x402b3160
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <component_ref 0x402b3120 type <integer_type 0x401da380 int>
        arg 0 <indirect_ref 0x402b2af0
            type <record_type 0x4026be00 _IO_FILE type_0 BLK
                size <integer_cst 0x40288400 constant 1184>
                unit size <integer_cst 0x402883c0 constant 148>
                align 32 symtab 0 alias set -1
                fields <field_decl 0x40284850 _flags
                type <integer_type 0x401da380 int>
                    in_system_header SI file /usr/include/libio.h line 262
                    size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 offset_align 128
                    offset <integer_cst 0x401ed5c0 constant 0>
                    bit offset <integer_cst 0x401ed680 constant 0>
                    context <record_type 0x4026be00 _IO_FILE>
                    arguments <integer_cst 0x401ed5c0 0>
                    chain <field_decl 0x402848c0 _IO_read_ptr>>
                pointer_to_this <pointer_type 0x40284460>
                chain <type_decl 0x4026be70>>

            arg 0 <parm_decl 0x402b19a0 __stream
                type <pointer_type 0x40292930
                    type <record_type 0x4026bf50 FILE BLK
                    size <integer_cst 0x40288400 1184>
                    unit size <integer_cst 0x402883c0 148>
                        align 32 symtab 0 alias set -1
                        fields <field_decl 0x40284850 _flags>
                        pointer_to_this <pointer_type 0x40292930>>
                    unsigned SI
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
                    align 32 symtab 0 alias set -1>
                unsigned used in_system_header SI
                file /usr/include/bits/stdio.h line 112
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402ab930 feof_unlocked>
                result <pointer_type 0x40292930>
```

```
                initial <pointer_type 0x40292930>
                arg-type <pointer_type 0x40292930>
                arg-type-as-written <pointer_type 0x40292930>>>
            arg 1 <field_decl 0x40284850 _flags>>
    arg 1 <integer_cst 0x402b3140 type <integer_type 0x401da380 int>
        constant 16>>
<bit_and_expr 0x402b3240
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <component_ref 0x402b3200 type <integer_type 0x401da380 int>
        arg 0 <indirect_ref 0x402b2d0c
            type <record_type 0x4026be00 _IO_FILE type_0 BLK
                size <integer_cst 0x40288400 constant 1184>
                unit size <integer_cst 0x402883c0 constant 148>
                align 32 symtab 0 alias set -1
                fields <field_decl 0x40284850 _flags
                type <integer_type 0x401da380 int>
                    in_system_header SI
                    file /usr/include/libio.h line 262
                    size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 offset_align 128
                    offset <integer_cst 0x401ed5c0 constant 0>
                    bit offset <integer_cst 0x401ed680 constant 0>
                    context <record_type 0x4026be00 _IO_FILE>
                    arguments <integer_cst 0x401ed5c0 0>
                    chain <field_decl 0x402848c0 _IO_read_ptr>>
                pointer_to_this <pointer_type 0x40284460>
                chain <type_decl 0x4026be70>>
            arg 0 <parm_decl 0x402b1b60 __stream
                type <pointer_type 0x40292930
                    type <record_type 0x4026bf50 FILE BLK
                        size <integer_cst 0x40288400 1184>
                        unit size <integer_cst 0x402883c0 148>
                        align 32 symtab 0 alias set -1
                        fields <field_decl 0x40284850 _flags>
                        pointer_to_this <pointer_type 0x40292930>>
                    unsigned SI
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
                    align 32 symtab 0 alias set -1>
                unsigned used in_system_header SI
                file /usr/include/bits/stdio.h line 119
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402aba80 ferror_unlocked>
                result <pointer_type 0x40292930>
```

```
                initial <pointer_type 0x40292930>
                arg-type <pointer_type 0x40292930>
                arg-type-as-written <pointer_type 0x40292930>>>
            arg 1 <field_decl 0x40284850 _flags>>
        arg 1 <integer_cst 0x402b3220 type <integer_type 0x401da380 int>
            constant 32>>
```

BIT_ANDTC_EXPR 2 2

bit 관련 연산들. Operand 들은 결과로써 같은 mode 를 가진다.

C 에서 사용되지 않는 TREE node 입니다.

BIT_NOT_EXPR 1 1

bit 관련 연산들. Operand 들은 결과로써 같은 mode 를 가진다.

```
 <bit_not_expr 0x401e549c
    type <integer_type 0x401da150 signed char QI
        size <integer_cst 0x401d73e0 constant 8>
        unit size <integer_cst 0x401d7400 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401d73a0 -128> max <integer_cst 0x401d73c0 127>>

    arg 0 <nop_expr 0x401e5488 type <integer_type 0x401da150 signed char>
        arg 0 <var_decl 0x401f6540 ch
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401d73e0 8>
                unit size <integer_cst 0x401d7400 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401d74a0 -128>
                max <integer_cst 0x401d74c0 127>
                pointer_to_this <pointer_type 0x401dec40>>
            asm_written used public static common QI file <stdin> line 1
            size <integer_cst 0x401d73e0 8>
            unit size <integer_cst 0x401d7400 1>
            align 8 initial <integer_cst 0x401e05e0 -61>
            (mem/f:QI (symbol_ref:SI ("ch")) [0 ch+0 S1 A8])
            chain <type_decl 0x401eed90 __g77_ulongint>>>>
```

TRUTH_ANDIF_EXPR e 2

ANDIF 와 ORIF 는 두번째 operand 를 허락하지만 만약 표현식의 값이 첫번째 operand 로부터 결정된다면 계산되지 않는다. AND 와 OR, XOR 은 두번째 operand 의 값이 필요하던 안하던 (부차적 효과를 위해) 두번째 operand 를 계산한다. Operand 는 BOOLEAN_TYPE 혹은 INTEGER_TYPE 를 가진다. 양쪽의 견우, Argument 는 0 혹은 1 일 것이다. 예를 들면, TRUTH_NOT_EXPR 는 그것의 argument 로써 절대 INTEGER_TYPE VAR_DECL 를 가지지 않을 것이다; 대신에, NE_EXPR 는 VAR_DECL 를 0 과 비교하는데 사용될 것이며, 그것에 의해서 값 0 혹은 1 을 가지는 node 를 가지게 된다.

```
 <truth_andif_expr 0x402d5240
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
```

```
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <ne_expr 0x402d51a0 type <integer_type 0x401da380 int>
        arg 0 <indirect_ref 0x402d2e10
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401eada0 constant 8>
                unit size <integer_cst 0x401eadc0 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128>
                max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401eec40>>
            readonly
            arg 0 <plus_expr 0x402d50e0
                type <pointer_type 0x401f1770
                    type <integer_type 0x401f1700 char readonly QI
                        size <integer_cst 0x401eada0 8>
                        unit size <integer_cst 0x401eadc0 1>
                        align 8 symtab 0 alias set -1 precision 8
                        min <integer_cst 0x401eae60 -128>
                        max <integer_cst 0x401eae80 127>
                        pointer_to_this <pointer_type 0x401f1770>>
                    unsigned SI
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
                    align 32 symtab 0 alias set -1
                    pointer_to_this <pointer_type 0x4027ee00>>
                arg 0 <parm_decl 0x402d3690 __s type <pointer_type 0x401f1770>
                    unsigned used in_system_header SI
                    file /usr/include/bits/string2.h line 919
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x402d3620 __strcspn_c1>
                    result <pointer_type 0x401f1770>
                    initial <pointer_type 0x401f1770>
                    arg-type <pointer_type 0x401f1770>
                    arg-type-as-written <pointer_type 0x401f1770>
                    chain <parm_decl 0x402d3700 __reject>>
                arg 1 <convert_expr 0x402d2dfc type <pointer_type 0x401f1770>
                    arg 0 <non_lvalue_expr 0x402d2de8
                        type <integer_type 0x402520e0 size_t unsigned SI
                            size <integer_cst 0x401eaf00 32>
                            unit size <integer_cst 0x401eafa0 4>
                            align 32 symtab 0 alias set -1 precision 32
                            min <integer_cst 0x401eafe0 0>
                            max <integer_cst 0x401ed000 4294967295>
                            pointer_to_this <pointer_type 0x40271c40>>
                        arg 0 <var_decl 0x402d38c0 __result
                            type <integer_type 0x402520e0 size_t>
                            unsigned used in_system_header common regdecl SI
```

```
                        file /usr/include/bits/string2.h line 921
                        size <integer_cst 0x401eaf00 32>
                        unit size <integer_cst 0x401eafa0 4>
                        align 32
                        context <function_decl 0x402d3620 __strcspn_c1>
                        initial <integer_cst 0x402d5060 0>>>>>>
        arg 1 <integer_cst 0x402d5180 constant 0>>
    arg 1 <ne_expr 0x402d5220 type <integer_type 0x401da380 int>
        arg 0 <nop_expr 0x402d2eec type <integer_type 0x401da380 int>
            readonly
            arg 0 <indirect_ref 0x402d2ed8 type <integer_type 0x401da230 char>
                readonly
                arg 0 <plus_expr 0x402d5200 type <pointer_type 0x401f1770>
                    arg 0 <parm_decl 0x402d3690 __s>
                    arg 1 <convert_expr 0x402d2ec4 type <pointer_type 0x401f1770>
                        arg 0 <non_lvalue_expr 0x402d2eb0
                            type <integer_type 0x402520e0 size_t>
                            arg 0 <var_decl 0x402d38c0 __result>>>>>>
        arg 1 <parm_decl 0x402d3700 __reject type <integer_type 0x401da380 int>
            used in_system_header SI file /usr/include/bits/string2.h line 919
            size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402d3620 __strcspn_c1>
            result <integer_type 0x401da380 int>
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>>>>
```

TRUTH_ORIF_EXPR e 2

ANDIF 와 ORIF 는 두번째 operand 를 허락하지만 만약 표현식의 값이 첫번째 operand 로부터 결정된다면 계산되지 않는다. AND 와 OR, XOR 은 두번째 operand 의 값이 필요하던 안하던 (부차적 효과를 위해) 두번째 operand 를 계산한다. Operand 는 BOOLEAN_TYPE 혹은 INTEGER_TYPE 를 가진다. 양쪽의 견우, Argument 는 0 혹은 1 일 것이다. 예를 들면, TRUTH_NOT_EXPR 는 그것의 argument 로써 절대 INTEGER_TYPE VAR_DECL 를 가지지 않을 것이다; 대신에, NE_EXPR 는 VAR_DECL 를 0 과 비교하는데 사용될 것이며, 그것에 의해서 값 0 혹은 1 을 가지는 node 를 가지게 된다.

```
 <truth_orif_expr 0x402d5d40
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <eq_expr 0x402d5ca0 type <integer_type 0x401da380 int>
        arg 0 <nop_expr 0x402d9dd4 type <integer_type 0x401da380 int>
            readonly
            arg 0 <indirect_ref 0x402d9dc0
                type <integer_type 0x401da230 char QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set -1 precision 8
```

```
                    min <integer_cst 0x401eae60 -128>
                    max <integer_cst 0x401eae80 127>
                    pointer_to_this <pointer_type 0x401eec40>>
                readonly
                arg 0 <plus_expr 0x402d5c80
                    type <pointer_type 0x401f1770
                        type <integer_type 0x401f1700 char readonly QI
                            size <integer_cst 0x401eada0 8>
                            unit size <integer_cst 0x401eadc0 1>
                            align 8 symtab 0 alias set -1 precision 8
                            min <integer_cst 0x401eae60 -128>
                            max <integer_cst 0x401eae80 127>
                            pointer_to_this <pointer_type 0x401f1770>>
                        unsigned SI
                        size <integer_cst 0x401ed540 constant 32>
                        unit size <integer_cst 0x401ed5a0 constant 4>
                        align 32 symtab 0 alias set -1
                        pointer_to_this <pointer_type 0x4027ee00>>
                    arg 0 <parm_decl 0x402d8b60 __s type <pointer_type 0x401f1770>
                        unsigned used in_system_header SI
                        file /usr/include/bits/string2.h line 988
                        size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 context <function_decl 0x402d8af0 __strspn_c2>
                        result <pointer_type 0x401f1770>
                        initial <pointer_type 0x401f1770>
                        arg-type <pointer_type 0x401f1770>
                        arg-type-as-written <pointer_type 0x401f1770>
                        chain <parm_decl 0x402d8bd0 __accept1>>
                    arg 1 <convert_expr 0x402d9dac type <pointer_type 0x401f1770>
                        arg 0 <non_lvalue_expr 0x402d9d98
                            type <integer_type 0x402520e0 size_t unsigned SI
                                size <integer_cst 0x401eaf00 32>
                                unit size <integer_cst 0x401eafa0 4>
                                align 32 symtab 0 alias set -1 precision 32
                                min <integer_cst 0x401eafe0 0>
                                max <integer_cst 0x401ed000 4294967295>
                                pointer_to_this <pointer_type 0x40271c40>>
                            arg 0 <var_decl 0x402d8e00 __result
                                type <integer_type 0x402520e0 size_t>
                                unsigned used in_system_header common regdecl SI
                                file /usr/include/bits/string2.h line 990
                                size <integer_cst 0x401eaf00 32>
                                unit size <integer_cst 0x401eafa0 4>
                                align 32
                                context <function_decl 0x402d8af0 __strspn_c2>
                                initial <integer_cst 0x402d5c00 0>>>>>>>
            arg 1 <parm_decl 0x402d8bd0 __accept1 type <integer_type 0x401da380 int>
                used in_system_header SI file /usr/include/bits/string2.h line 988
                size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
                align 32 context <function_decl 0x402d8af0 __strspn_c2>
                result <integer_type 0x401da380 int>
```

```
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>
            chain <parm_decl 0x402d8c40 __accept2>>>
    arg 1 <eq_expr 0x402d5d20 type <integer_type 0x401da380 int>
        arg 0 <nop_expr 0x402d9e4c type <integer_type 0x401da380 int>
            readonly
            arg 0 <indirect_ref 0x402d9e38 type <integer_type 0x401da230 char>
                readonly
                arg 0 <plus_expr 0x402d5d00 type <pointer_type 0x401f1770>
                    arg 0 <parm_decl 0x402d8b60 __s>
                    arg 1 <convert_expr 0x402d9e24 type <pointer_type 0x401f1770>
                        arg 0 <non_lvalue_expr 0x402d9e10
                            type <integer_type 0x402520e0 size_t>
                            arg 0 <var_decl 0x402d8e00 __result>>>>>>
        arg 1 <parm_decl 0x402d8c40 __accept2 type <integer_type 0x401da380 int>
            used in_system_header SI file /usr/include/bits/string2.h line 988
            size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402d8af0 __strspn_c2>
            result <integer_type 0x401da380 int>
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>>>>
<truth_orif_expr 0x402f2820
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <truth_orif_expr 0x402f27e0 type <integer_type 0x401da380 int>
        arg 0 <eq_expr 0x402f27a0 type <integer_type 0x401da380 int>
            arg 0 <indirect_ref 0x402f55f0
                type <integer_type 0x401da230 char QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set -1 precision 8
                    min <integer_cst 0x401eae60 -128>
                    max <integer_cst 0x401eae80 127>
                    pointer_to_this <pointer_type 0x401eec40>>
                arg 0 <var_decl 0x402f4850 __cp
                    type <pointer_type 0x401eec40
                        type <integer_type 0x401da230 char>
                        unsigned SI
                        size <integer_cst 0x401ed540 constant 32>
                        unit size <integer_cst 0x401ed5a0 constant 4>
                        align 32 symtab 0 alias set -1
                        pointer_to_this <pointer_type 0x402979a0>>
                    unsigned used in_system_header common regdecl SI
                    file /usr/include/bits/string2.h line 1178
                    size <integer_cst 0x401ed540 32>
```

```
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x402f4460 __strsep_3c>
                    initial <var_decl 0x402f47e0 __retval>>>
            arg 1 <parm_decl 0x402f4540 __reject1
                type <integer_type 0x401da230 char>
                used in_system_header QI file /usr/include/bits/string2.h line 1173
                size <integer_cst 0x401eada0 8>
                unit size <integer_cst 0x401eadc0 1>
                align 8 context <function_decl 0x402f4460 __strsep_3c>
                result <integer_type 0x401da230 char>
                initial <integer_type 0x401da380 int>
                arg-type <integer_type 0x401da380 int>
                arg-type-as-written <integer_type 0x401da230 char>
                chain <parm_decl 0x402f45b0 __reject2>>>
        arg 1 <eq_expr 0x402f27c0 type <integer_type 0x401da380 int>
            arg 0 <indirect_ref 0x402f562c type <integer_type 0x401da230 char>
                arg 0 <var_decl 0x402f4850 __cp>>
            arg 1 <parm_decl 0x402f45b0 __reject2
                type <integer_type 0x401da230 char>
                used in_system_header QI file /usr/include/bits/string2.h line 1173
                size <integer_cst 0x401eada0 8>
                unit size <integer_cst 0x401eadc0 1>
                align 8 context <function_decl 0x402f4460 __strsep_3c>
                result <integer_type 0x401da230 char>
                initial <integer_type 0x401da380 int>
                arg-type <integer_type 0x401da380 int>
                arg-type-as-written <integer_type 0x401da230 char>
                chain <parm_decl 0x402f4620 __reject3>>>>
    arg 1 <eq_expr 0x402f2800 type <integer_type 0x401da380 int>
        arg 0 <indirect_ref 0x402f56b8 type <integer_type 0x401da230 char>
            arg 0 <var_decl 0x402f4850 __cp>>
        arg 1 <parm_decl 0x402f4620 __reject3 type <integer_type 0x401da230 char>
            used in_system_header QI file /usr/include/bits/string2.h line 1173
            size <integer_cst 0x401eada0 8> unit size <integer_cst 0x401eadc0 1>
            align 8 context <function_decl 0x402f4460 __strsep_3c>
            result <integer_type 0x401da230 char>
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da230 char>>>>
```

TRUTH_AND_EXPR e 2
TRUTH_OR_EXPR e 2

ANDIF 와 ORIF 는 두번째 operand 를 허락하지만 만약 표현식의 값이 첫번째 operand 로부터 결정된다면 계산되지 않는다. AND 와 OR, XOR 은 두번째 operand 의 값이 필요하던 안하던 (부차적 효과를 위해) 두번째 operand 를 계산한다. Operand 는 BOOLEAN_TYPE 혹은 INTEGER_TYPE 를 가진다. 양쪽의 견우, Argument 는 0 혹은 1 일 것이다. 예를 들면, TRUTH_NOT_EXPR 는 그것의 argument 로써 절대 INTEGER_TYPE VAR_DECL 를 가지지 않을 것이다; 대신에, NE_EXPR 는 VAR_DECL 를 0 과 비교하는데 사용될 것이며, 그것에 의해서 값 0 혹은 1 을 가지는 node 를 가지게 된다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

TRUTH_XOR_EXPR e 2

ANDIF 와 ORIF 는 두번째 operand 를 허락하지만 만약 표현식의 값이 첫번째 operand 로부터 결정된다면 계산되지 않는다. AND 와 OR, XOR 은 두번째 operand 의 값이 필요하던 안하던 (부차적 효과를 위해) 두번째 operand 를 계산한다. Operand 는 BOOLEAN_TYPE 혹은 INTEGER_TYPE 를 가진다. 양쪽의 견우, Argument 는 0 혹은 1 일 것이다. 예를 들면, TRUTH_NOT_EXPR 는 그것의 argument 로써 절대 INTEGER_TYPE VAR_DECL 를 가지지 않을 것이다; 대신에, NE_EXPR 는 VAR_DECL 를 0 과 비교하는데 사용될 것이며, 그것에 의해서 값 0 혹은 1 을 가지는 node 를 가지게 된다.

```
<truth_xor_expr 0x40281060
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       imin <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   arg 0 <ge_expr 0x40281040 type <integer_type 0x401da380 int>
       arg 0 <var_decl 0x4027f540 d
           type <integer_type 0x40272bd0 slongest_t DI
               size <integer_cst 0x401d7600 constant 64>
               unit size <integer_cst 0x401d7720 constant 8>
               align 64 symtab 0 alias set -1 precision 64
               min <integer_cst 0x401d76e0 0x8000000000000000>
               max <integer_cst 0x401d7700 0x7fffffffffffffff>>
           used common DI file <stdin> line 119 size <integer_cst
           0x401d7600 64>
           unit size <integer_cst 0x401d7720 8>
           align 64 context <function_decl 0x40276380 Check_Signed>>
       arg 1 <integer_cst 0x4027ef80 constant 0>>
   arg 1 <lt_expr 0x4027efe0 type <integer_type 0x401da380 int>
       arg 0 <parm_decl 0x40276b60 n
           type <integer_type 0x402759a0 long long int readonly DI
               size <integer_cst 0x401d7600 64>
               unit size <integer_cst 0x401d7720 8>
               align 64 symtab 0 alias set -1 precision 64
               min <integer_cst 0x401d76e0 0x8000000000000000>
               max <integer_cst 0x401d7700 0x7fffffffffffffff>
               pointer_to_this <pointer_type 0x40275e00>>
           readonly used DI file <stdin> line 101
           size <integer_cst 0x401d7600 64> unit size <integer_cst 0x401d7720 8>
           align 64 context <function_decl 0x40276380 Check_Signed>
           result <integer_type 0x402759a0 long long int>
           initial <integer_type 0x402759a0 long long int>
           arg-type <integer_type 0x402759a0 long long int>
           arg-type-as-written <integer_type 0x402759a0 long long int>
           chain <parm_decl 0x40276bd0 dens>>
       arg 1 <integer_cst 0x4027efc0 constant 0>>>
<truth_xor_expr 0x40281020
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
```

```
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    arg 0 <lt_expr 0x4027efa0 type <integer_type 0x401da380 int>
        arg 0 <var_decl 0x4027f540 d
            type <integer_type 0x40272bd0 slongest_t DI
                size <integer_cst 0x401d7600 constant 64>
                unit size <integer_cst 0x401d7720 constant 8>
                align 64 symtab 0 alias set -1 precision 64
                min <integer_cst 0x401d76e0 0x8000000000000000>
                max <integer_cst 0x401d7700 0x7fffffffffffffff>>
            used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
            unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>>
        arg 1 <integer_cst 0x4027ef80 constant 0>>
    arg 1 <lt_expr 0x4027efe0 type <integer_type 0x401da380 int>
        arg 0 <parm_decl 0x40276b60 n
            type <integer_type 0x402759a0 long long int readonly DI
                size <integer_cst 0x401d7600 64>
                unit size <integer_cst 0x401d7720 8>
                align 64 symtab 0 alias set -1 precision 64
                min <integer_cst 0x401d76e0 0x8000000000000000>
                max <integer_cst 0x401d7700 0x7fffffffffffffff>
                pointer_to_this <pointer_type 0x40275e00>>
            readonly used DI file <stdin> line 101
            size <integer_cst 0x401d7600 64> unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>
            result <integer_type 0x402759a0 long long int>
            initial <integer_type 0x402759a0 long long int>
            arg-type <integer_type 0x402759a0 long long int>
            arg-type-as-written <integer_type 0x402759a0 long long int>
            chain <parm_decl 0x40276bd0 dens>>
        arg 1 <integer_cst 0x4027efc0 constant 0>>>
```

TRUTH_NOT_EXPR e 1

ANDIF 와 ORIF 는 두번째 operand 를 허락하지만 만약 표현식의 값이 첫번째 operand 로부터 결정된다면 계산되지 않는다. AND 와 OR, XOR 은 두번째 operand 의 값이 필요하던 안하던 (부차적 효과를 위해) 두번째 operand 를 계산한다. Operand 는 BOOLEAN_TYPE 혹은 INTEGER_TYPE 를 가진다. 양쪽의 견우, Argument 는 0 혹은 1 일 것이다. 예를 들면, TRUTH_NOT_EXPR 는 그것의 argument 로써 절대 INTEGER_TYPE VAR_DECL 를 가지지 않을 것이다; 대신에, NE_EXPR 는 VAR_DECL 를 0 과 비교하는데 사용될 것이며, 그것에 의해서 값 0 혹은 1 을 가지는 node 를 가지게 된다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

LT_EXPR < 2

```
 <lt_expr 0x40281800
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
```

```
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    arg 0 <var_decl 0x4027f690 r
        type <integer_type 0x40272bd0 slongest_t DI
            size <integer_cst 0x401d7600 constant 64>
            unit size <integer_cst 0x401d7720 constant 8>
            align 64 symtab 0 alias set -1 precision 64
            min <integer_cst 0x401d76e0 0x8000000000000000>
            max <integer_cst 0x401d7700 0x7fffffffffffffff>>
        used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
        unit size <integer_cst 0x401d7720 8>
        align 64 context <function_decl 0x40276380 Check_Signed>
        chain <var_decl 0x4027f620 q type <integer_type 0x40272bd0 slongest_t>
            used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
            unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>
            chain <var_decl 0x4027f5b0 dneg>>>
    arg 1 <integer_cst 0x402817e0 type <integer_type 0x40272bd0 slongest_t>
        constant 0>>
<lt_expr 0x40281740
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    arg 0 <parm_decl 0x40276b60 n
        type <integer_type 0x402759a0 long long int readonly DI
            size <integer_cst 0x401d7600 constant 64>
            unit size <integer_cst 0x401d7720 constant 8>
            align 64 symtab 0 alias set -1 precision 64
            min <integer_cst 0x401d76e0 0x8000000000000000>
            max <integer_cst 0x401d7700 0x7fffffffffffffff>
            pointer_to_this <pointer_type 0x40275e00>>
        readonly used DI file <stdin> line 101 size <integer_cst 0x401d7600 64>
        unit size <integer_cst 0x401d7720 8>
        align 64 context <function_decl 0x40276380 Check_Signed>
        result <integer_type 0x402759a0 long long int>
        initial <integer_type 0x402759a0 long long int>
        arg-type <integer_type 0x402759a0 long long int>
        arg-type-as-written <integer_type 0x402759a0 long long int>
        chain <parm_decl 0x40276bd0 dens
            type <pointer_type 0x40275cb0
                type <integer_type 0x40275700 long long unsigned int readonly
                    unsigned DI size <integer_cst 0x401d7600 64>
                    unit size <integer_cst 0x401d7720 8>
                    align 64 symtab 0 alias set -1 precision 64
                    min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>
```

```
                    pointer_to_this <pointer_type 0x40275c40>>
                readonly unsigned SI
                size <integer_cst 0x401d7b80 constant 32>
                unit size <integer_cst 0x401d7be0 constant 4>
                align 32 symtab 0 alias set -1>
            readonly unsigned used SI file <stdin> line 102
            size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
            align 32 context <function_decl 0x40276380 Check_Signed>
            result <pointer_type 0x40275cb0> initial <pointer_type 0x40275cb0>
            arg-type <pointer_type 0x40275cb0>
            arg-type-as-written <pointer_type 0x40275cb0>
            chain <parm_decl 0x40276c40 smax>>>
        arg 1 <integer_cst 0x40281720 type <integer_type 0x402759a0 long long int>
            constant 0>>
```

LE_EXPR < 2

관계 연산자들.
'EQ_EXPR' 와 'NE_EXPR' 는 어떠한 type 들도 허락한다.
다른 것들은 오직 정수 (혹은 포인터 혹은 열거자) 혹은 실수 type 들에 대해서만 허락한다.
모든 경우에 대해 operand 는 같은 type 을 가질 것이고, 값은 항상 Boolean 에 대해 언어가 사용하는 type 일 것이다.

```
 <le_expr 0x40281820
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    arg 0 <negate_expr 0x4028249c
        type <integer_type 0x40272bd0 slongest_t DI
            size <integer_cst 0x401d7600 constant 64>
            unit size <integer_cst 0x401d7720 constant 8>
            align 64 symtab 0 alias set -1 precision 64
            min <integer_cst 0x401d76e0 0x8000000000000000>
            max <integer_cst 0x401d7700 0x7fffffffffffffff>>
        arg 0 <var_decl 0x4027f690 r type <integer_type 0x40272bd0 slongest_t>
            used common DI file <stdin> line 119
            size <integer_cst 0x401d7600 64>
            unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>
            chain <var_decl 0x4027f620 q>>>
    arg 1 <var_decl 0x4027f5b0 dneg type <integer_type 0x40272bd0 slongest_t>
        used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
        unit size <integer_cst 0x401d7720 8>
        align 64 context <function_decl 0x40276380 Check_Signed>
        chain <var_decl 0x4027f540 d type <integer_type 0x40272bd0 slongest_t>
            used common DI file <stdin> line 119
            size <integer_cst 0x401d7600 64>
            unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>>>>
```

```
<le_expr 0x402817a0
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   arg 0 <var_decl 0x4027f690 r
       type <integer_type 0x40272bd0 slongest_t DI
           size <integer_cst 0x401d7600 constant 64>
           unit size <integer_cst 0x401d7720 constant 8>
           align 64 symtab 0 alias set -1 precision 64
           min <integer_cst 0x401d76e0 0x8000000000000000>
           max <integer_cst 0x401d7700 0x7fffffffffffffff>>
       used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
       unit size <integer_cst 0x401d7720 8>
       align 64 context <function_decl 0x40276380 Check_Signed>
       chain <var_decl 0x4027f620 q type <integer_type 0x40272bd0 slongest_t>
           used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
           unit size <integer_cst 0x401d7720 8>
           align 64 context <function_decl 0x40276380 Check_Signed>
           chain <var_decl 0x4027f5b0 dneg>>>
   arg 1 <var_decl 0x4027f5b0 dneg type <integer_type 0x40272bd0 slongest_t>
       used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
       unit size <integer_cst 0x401d7720 8>
       align 64 context <function_decl 0x40276380 Check_Signed>
       chain <var_decl 0x4027f540 d type <integer_type 0x40272bd0 slongest_t>
           used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
           unit size <integer_cst 0x401d7720 8>
           align 64 context <function_decl 0x40276380 Check_Signed>>>>
```

GT_EXPR <

관계 연산자들.
'EQ_EXPR' 와 'NE_EXPR' 는 어떠한 type 들도 허락한다.
다른 것들은 오직 정수 (혹은 포인터 혹은 열거자) 혹은 실수 type 들에 대해서만 허락한다.
모든 경우에 대해 operand 는 같은 type 을 가질 것이고, 값은 항상 Boolean 에 대해 언어가 사용하는 type 일 것이다.

```
<gt_expr 0x40281780
   type <integer_type 0x401da380 int SI
       size <integer_cst 0x401d7540 constant 32>
       unit size <integer_cst 0x401d75e0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401d75a0 -2147483648>
       max <integer_cst 0x401d75c0 2147483647>
       pointer_to_this <pointer_type 0x401e2620>>
   arg 0 <var_decl 0x4027f690 r
       type <integer_type 0x40272bd0 slongest_t DI
           size <integer_cst 0x401d7600 constant 64>
           unit size <integer_cst 0x401d7720 constant 8>
           align 64 symtab 0 alias set -1 precision 64
```

```
            min <integer_cst 0x401d76e0 0x8000000000000000>
            max <integer_cst 0x401d7700 0x7fffffffffffffff>>
        used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
        unit size <integer_cst 0x401d7720 8>
        align 64 context <function_decl 0x40276380 Check_Signed>
        chain <var_decl 0x4027f620 q type <integer_type 0x40272bd0 slongest_t>
            used common DI file <stdin> line 119 size <integer_cst 0x401d7600 64>
            unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>
            chain <var_decl 0x4027f5b0 dneg>>>
    arg 1 <integer_cst 0x40281760 type <integer_type 0x40272bd0 slongest_t>
        constant 0>>
<gt_expr 0x402812c0
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    readonly
    arg 0 <parm_decl 0x40276b60 n
        type <integer_type 0x402759a0 long long int readonly DI
            size <integer_cst 0x401d7600 constant 64>
            unit size <integer_cst 0x401d7720 constant 8>
            align 64 symtab 0 alias set -1 precision 64
            min <integer_cst 0x401d76e0 0x8000000000000000>
            max <integer_cst 0x401d7700 0x7fffffffffffffff>
            pointer_to_this <pointer_type 0x40275e00>>
        readonly used DI file <stdin> line 101 size <integer_cst 0x401d7600 64>
        unit size <integer_cst 0x401d7720 8>
        align 64 context <function_decl 0x40276380 Check_Signed>
        result <integer_type 0x402759a0 long long int>
        initial <integer_type 0x402759a0 long long int>
        arg-type <integer_type 0x402759a0 long long int>
        arg-type-as-written <integer_type 0x402759a0 long long int>
        chain <parm_decl 0x40276bd0 dens
            type <pointer_type 0x40275cb0
                type <integer_type 0x40275700 long long unsigned int readonly
                    unsigned DI size <integer_cst 0x401d7600 64>
                    unit size <integer_cst 0x401d7720 8>
                    align 64 symtab 0 alias set -1 precision 64
                    min <integer_cst 0x401d7760 0> max <integer_cst 0x401d7780 -1>
                    pointer_to_this <pointer_type 0x40275c40>>
                readonly unsigned SI
                size <integer_cst 0x401d7b80 constant 32>
                unit size <integer_cst 0x401d7be0 constant 4>
                align 32 symtab 0 alias set -1>
            readonly unsigned used SI file <stdin> line 102
            size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
            align 32 context <function_decl 0x40276380 Check_Signed>
            result <pointer_type 0x40275cb0> initial <pointer_type 0x40275cb0>
```

```
            arg-type <pointer_type 0x40275cb0>
            arg-type-as-written <pointer_type 0x40275cb0>
            chain <parm_decl 0x40276c40 smax>>>
        arg 1 <var_decl 0x4027fa10 dsign type <integer_type 0x402759a0 long long int>
            readonly used common DI file <stdin> line 157
            size <integer_cst 0x401d7600 64> unit size <integer_cst 0x401d7720 8>
            align 64 context <function_decl 0x40276380 Check_Signed>
            initial <cond_expr 0x402812a0>>>
```

GE_EXPR < 2

관계 연산자들.
'EQ_EXPR' 와 'NE_EXPR' 는 어떠한 type 들도 허락한다.
다른 것들은 오직 정수 (혹은 포인터 혹은 열거자) 혹은 실수 type 들에 대해서만 허락한다.
모든 경우에 대해 operand 는 같은 type 을 가질 것이고, 값은 항상 Boolean 에 대해 언어가 사용하는 type 일 것이다.

```
 <ge_expr 0x402a0c20
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <component_ref 0x402a0be0
        type <pointer_type 0x401eec40
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401eada0 constant 8>
                unit size <integer_cst 0x401eadc0 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401eec40>>
            unsigned SI
            size <integer_cst 0x401ed540 constant 32>
            unit size <integer_cst 0x401ed5a0 constant 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x402979a0>>
        arg 0 <indirect_ref 0x402b0924
            type <record_type 0x4026be00 _IO_FILE type_0 BLK
                size <integer_cst 0x40288400 constant 1184>
                unit size <integer_cst 0x402883c0 constant 148>
                align 32 symtab 0 alias set -1
                fields <field_decl 0x40284850 _flags
                    type <integer_type 0x401da380 int>
                    in_system_header SI file /usr/include/libio.h line 262
                    size <integer_cst 0x401eaf00 32>
                    unit size <integer_cst 0x401eafa0 4>
                    align 32 offset_align 128
                    offset <integer_cst 0x401ed5c0 constant 0>
                    bit offset <integer_cst 0x401ed680 constant 0>
                    context <record_type 0x4026be00 _IO_FILE>
                    arguments <integer_cst 0x401ed5c0 0>
```

```
                    chain <field_decl 0x402848c0 _IO_read_ptr>>
                pointer_to_this <pointer_type 0x40284460>
                chain <type_decl 0x4026be70>>
            arg 0 <parm_decl 0x402aff50 __stream
                type <pointer_type 0x40292930
                    type <record_type 0x4026bf50 FILE BLK
                        size <integer_cst 0x40288400 1184>
                        unit size <integer_cst 0x402883c0 148>
                        align 32 symtab 0 alias set -1
                        fields <field_decl 0x40284850 _flags>
                        pointer_to_this <pointer_type 0x40292930>>
                    unsigned SI size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 symtab 0 alias set -1>
                unsigned used in_system_header SI
                file /usr/include/bits/stdio.h line 75
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402a24d0 fputc_unlocked>
                result <pointer_type 0x40292930>
                initial <pointer_type 0x40292930>
                arg-type <pointer_type 0x40292930>
                arg-type-as-written <pointer_type 0x40292930>>>
        arg 1 <field_decl 0x40284a80 _IO_write_ptr type <pointer_type 0x401eec40>
            unsigned in_system_header SI file /usr/include/libio.h line 271
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 offset_align 128
            offset <integer_cst 0x401ed7c0 constant 16>
            bit offset <integer_cst 0x401ed540 32>
            context <record_type 0x4026be00 _IO_FILE>
            arguments <integer_cst 0x401ed7c0 16>
            chain <field_decl 0x40284af0 _IO_write_end>>>
    arg 1 <component_ref 0x402a0c00 type <pointer_type 0x401eec40>
        arg 0 <indirect_ref 0x402b0960 type <record_type 0x4026be00 _IO_FILE>
            arg 0 <parm_decl 0x402aff50 __stream>>
        arg 1 <field_decl 0x40284af0 _IO_write_end type <pointer_type 0x401eec40>
            unsigned in_system_header SI file /usr/include/libio.h line 272
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 offset_align 128 offset <integer_cst 0x401ed7c0 16>
            bit offset <integer_cst 0x401ed2c0 constant 64>
            context <record_type 0x4026be00 _IO_FILE>
            arguments <integer_cst 0x401ed7c0 16>
            chain <field_decl 0x40284b60 _IO_buf_base>>>>
<ge_expr 0x402a0f20
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <component_ref 0x402a0ee0
```

```
type <pointer_type 0x401eec40
    type <integer_type 0x401da230 char QI
        size <integer_cst 0x401eada0 constant 8>
        unit size <integer_cst 0x401eadc0 constant 1>
        align 8 symtab 0 alias set -1 precision 8
        min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
        pointer_to_this <pointer_type 0x401eec40>>
    unsigned SI
    size <integer_cst 0x401ed540 constant 32>
    unit size <integer_cst 0x401ed5a0 constant 4>
    align 32 symtab 0 alias set -1
    pointer_to_this <pointer_type 0x402979a0>>
arg 0 <indirect_ref 0x402b226c
    type <record_type 0x4026be00 _IO_FILE type_0 BLK
        size <integer_cst 0x40288400 constant 1184>
        unit size <integer_cst 0x402883c0 constant 148>
        align 32 symtab 0 alias set -1
        fields <field_decl 0x40284850 _flags
            type <integer_type 0x401da380 int>
            in_system_header SI file /usr/include/libio.h line 262
            size <integer_cst 0x401eaf00 32>
            unit size <integer_cst 0x401eafa0 4>
            align 32 offset_align 128
            offset <integer_cst 0x401ed5c0 constant 0>
            bit offset <integer_cst 0x401ed680 constant 0>
            context <record_type 0x4026be00 _IO_FILE>
            arguments <integer_cst 0x401ed5c0 0>
            chain <field_decl 0x402848c0 _IO_read_ptr>>
        pointer_to_this <pointer_type 0x40284460>
        chain <type_decl 0x4026be70>>
    arg 0 <var_decl 0x40292a10 stdout
        type <pointer_type 0x40292930
            type <record_type 0x4026bf50 FILE BLK
                size <integer_cst 0x40288400 1184>
                unit size <integer_cst 0x402883c0 148>
                align 32 symtab 0 alias set -1
                fields <field_decl 0x40284850 _flags>
                pointer_to_this <pointer_type 0x40292930>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>
        unsigned used public in_system_header common external
        defer-output SI file include/stdio.h line 143
        size <integer_cst 0x401ed540 32>
        unit size <integer_cst 0x401ed5a0 4>
        align 32 chain <var_decl 0x402929a0 stdin>>>
arg 1 <field_decl 0x40284a80 _IO_write_ptr type <pointer_type 0x401eec40>
    unsigned in_system_header SI file /usr/include/libio.h line 271
    size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
    align 32 offset_align 128
    offset <integer_cst 0x401ed7c0 constant 16>
    bit offset <integer_cst 0x401ed540 32>
```

```
            context <record_type 0x4026be00 _IO_FILE>
            arguments <integer_cst 0x401ed7c0 16>
            chain <field_decl 0x40284af0 _IO_write_end>>>
    arg 1 <component_ref 0x402a0f00 type <pointer_type 0x401eec40>
        arg 0 <indirect_ref 0x402b22a8 type <record_type 0x4026be00 _IO_FILE>
            arg 0 <var_decl 0x40292a10 stdout>>
        arg 1 <field_decl 0x40284af0 _IO_write_end type <pointer_type 0x401eec40>
            unsigned in_system_header SI file /usr/include/libio.h line 272
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 offset_align 128 offset <integer_cst 0x401ed7c0 16>
            bit offset <integer_cst 0x401ed2c0 constant 64>
            context <record_type 0x4026be00 _IO_FILE>
            arguments <integer_cst 0x401ed7c0 16>
            chain <field_decl 0x40284b60 _IO_buf_base>>>>
```

EQ_EXPR < 2

관계 연산자들.
'EQ_EXPR' 와 'NE_EXPR' 는 어떠한 type 들도 허락한다.
다른 것들은 오직 정수 (혹은 포인터 혹은 열거자) 혹은 실수 type 들에 대해서만 허락한다.
모든 경우에 대해 operand 는 같은 type 을 가질 것이고, 값은 항상 Boolean 에 대해 언어가 사용하는 type 일 것이다.

```
 <eq_expr 0x402d5b20
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <nop_expr 0x402d9758 type <integer_type 0x401da380 int>
        readonly
        arg 0 <indirect_ref 0x402d9744
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401eada0 constant 8>
                unit size <integer_cst 0x401eadc0 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401eec40>>
            readonly
            arg 0 <plus_expr 0x402d5b00
                type <pointer_type 0x401f1770
                    type <integer_type 0x401f1700 char readonly QI
                        size <integer_cst 0x401eada0 8>
                        unit size <integer_cst 0x401eadc0 1>
                        align 8 symtab 0 alias set -1 precision 8
                        min <integer_cst 0x401eae60 -128>
                        max <integer_cst 0x401eae80 127>
                        pointer_to_this <pointer_type 0x401f1770>>
                    unsigned SI
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
```

```
                    align 32 symtab 0 alias set -1
                    pointer_to_this <pointer_type 0x4027ee00>>
                arg 0 <parm_decl 0x402d8690 __s type <pointer_type 0x401f1770>
                    unsigned used in_system_header SI
                    file /usr/include/bits/string2.h line 976
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x402d8620 __strspn_c1>
                    result <pointer_type 0x401f1770>
                    initial <pointer_type 0x401f1770>
                    arg-type <pointer_type 0x401f1770>
                    arg-type-as-written <pointer_type 0x401f1770>
                    chain <parm_decl 0x402d8700 __accept>>
                arg 1 <convert_expr 0x402d9730 type <pointer_type 0x401f1770>
                    arg 0 <non_lvalue_expr 0x402d971c
                        type <integer_type 0x402520e0 size_t unsigned SI
                            size <integer_cst 0x401eaf00 32>
                            unit size <integer_cst 0x401eafa0 4>
                            align 32 symtab 0 alias set -1 precision 32
                            min <integer_cst 0x401eafe0 0>
                            max <integer_cst 0x401ed000 4294967295>
                            pointer_to_this <pointer_type 0x40271c40>>
                        arg 0 <var_decl 0x402d88c0 __result
                            type <integer_type 0x402520e0 size_t>
                            unsigned used in_system_header common regdecl SI
                            file /usr/include/bits/string2.h line 978
                            size <integer_cst 0x401eaf00 32>
                            unit size <integer_cst 0x401eafa0 4>
                            align 32 context <function_decl 0x402d8620 __strspn_c1>
                            initial <integer_cst 0x402d5a80 0>>>>>>>
        arg 1 <parm_decl 0x402d8700 __accept type <integer_type 0x401da380 int>
            used in_system_header SI file /usr/include/bits/string2.h line 976
            size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402d8620 __strspn_c1>
            result <integer_type 0x401da380 int> initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>>>
    <eq_expr 0x402f2700
        type <integer_type 0x401da380 int SI
            size <integer_cst 0x401eaf00 constant 32>
            unit size <integer_cst 0x401eafa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401eaf60 -2147483648>
            max <integer_cst 0x401eaf80 2147483647>
            pointer_to_this <pointer_type 0x401f1620>>
        arg 0 <indirect_ref 0x402f549c
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401eada0 constant 8>
                unit size <integer_cst 0x401eadc0 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401eec40>>
```

```
            arg 0 <var_decl 0x402f4850 __cp
                type <pointer_type 0x401eec40 type <integer_type 0x401da230 char>
                    unsigned SI
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
                    align 32 symtab 0 alias set -1
                    pointer_to_this <pointer_type 0x402979a0>>
                unsigned used in_system_header common regdecl SI
                file /usr/include/bits/string2.h line 1178
                size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402f4460 __strsep_3c>
                initial <var_decl 0x402f47e0 __retval>>>
        arg 1 <integer_cst 0x402f26e0 type <integer_type 0x401da230 char> constant 0>>
```

NE_EXPR < 2

관계 연산자들.
'EQ_EXPR' 와 'NE_EXPR' 는 어떠한 type 들도 허락한다.
다른 것들은 오직 정수 (혹은 포인터 혹은 열거자) 혹은 실수 type 들에 대해서만 허락한다.
모든 경우에 대해 operand 는 같은 type 을 가질 것이고, 값은 항상 Boolean 에 대해 언어가 사용하는 type 일 것이다.

```
 <ne_expr 0x402b3180
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <bit_and_expr 0x402b3160 type <integer_type 0x401da380 int>
        arg 0 <component_ref 0x402b3120 type <integer_type 0x401da380 int>

            arg 0 <indirect_ref 0x402b2af0
                type <record_type 0x4026be00 _IO_FILE type_0 BLK
                    size <integer_cst 0x40288400 constant 1184>
                    unit size <integer_cst 0x402883c0 constant 148>
                    align 32 symtab 0 alias set -1
                    fields <field_decl 0x40284850 _flags
                        type <integer_type 0x401da380 int>
                        in_system_header SI file /usr/include/libio.h line 262
                        size <integer_cst 0x401eaf00 32>
                        unit size <integer_cst 0x401eafa0 4>
                        align 32 offset_align 128
                        offset <integer_cst 0x401ed5c0 constant 0>
                        bit offset <integer_cst 0x401ed680 constant 0>
                        context <record_type 0x4026be00 _IO_FILE>
                        arguments <integer_cst 0x401ed5c0 0>
                        chain <field_decl 0x402848c0 _IO_read_ptr>>
                    pointer_to_this <pointer_type 0x40284460>
                    chain <type_decl 0x4026be70>>
                arg 0 <parm_decl 0x402b19a0 __stream
                    type <pointer_type 0x40292930
```

```
                        type <record_type 0x4026bf50 FILE BLK
                            size <integer_cst 0x40288400 1184>
                            unit size <integer_cst 0x402883c0 148>
                            align 32 symtab 0 alias set -1
                            fields <field_decl 0x40284850 _flags>
                            pointer_to_this <pointer_type 0x40292930>>
                        unsigned SI
                        size <integer_cst 0x401ed540 constant 32>
                        unit size <integer_cst 0x401ed5a0 constant 4>
                        align 32 symtab 0 alias set -1>
                    unsigned used in_system_header SI
                    file /usr/include/bits/stdio.h line 112
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x402ab930 feof_unlocked>
                    result <pointer_type 0x40292930>
                    initial <pointer_type 0x40292930>
                    arg-type <pointer_type 0x40292930>
                    arg-type-as-written <pointer_type 0x40292930>>>
            arg 1 <field_decl 0x40284850 _flags>>
        arg 1 <integer_cst 0x402b3140 constant 16>>
    arg 1 <integer_cst 0x401ed620 type <integer_type 0x401da380 int> constant 0>>
<ne_expr 0x402ddb80
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eaf60 -2147483648>
        max <integer_cst 0x401eaf80 2147483647>
        pointer_to_this <pointer_type 0x401f1620>>
    arg 0 <indirect_ref 0x402f13c0
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        arg 0 <parm_decl 0x402f01c0 __s
            type <pointer_type 0x401eec40 type <integer_type 0x401da230 char>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x402979a0>>
            unsigned used in_system_header SI
            file /usr/include/bits/string2.h line 1085
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402f0150 __strtok_r_1c>
            result <pointer_type 0x401eec40> initial <pointer_type 0x401eec40>
            arg-type <pointer_type 0x401eec40>
            arg-type-as-written <pointer_type 0x401eec40>
            chain <parm_decl 0x402f0230 __sep>>>
```

```
arg 1 <integer_cst 0x402ddb60 type <integer_type 0x401da230 char> constant 0>>
```

UNORDERED_EXPR < 2

Unordered 인 부동소수점을 위한 추가적인 관계 연산자들.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

ORDERED_EXPR < 2

Unordered 인 부동소수점을 위한 추가적인 관계 연산자들.

C 에서 사용되지 않는 TREE node 입니다.

UNLT_EXPR < 2
UNLE_EXPR < 2
UNGT_EXPR < 2
UNGE_EXPR < 2
UNEQ_EXPR < 2

다음은 unordered 혹은 ... 와 같다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다. 적절한 예제를 찾으시는 분은 저에게 메일로 보내주시기 바랍니다.

IN_EXPR 2 2
SET_LE_EXPR < 2
CARD_EXPR 1 1
RANGE_EXPR 2 2

Pascal 집합들을 위한 연산자들, 이제 사용되지 않는다.

C 에서 사용되지 않는 TREE node 들 입니다.

CONVERT_EXPR 1 1

값의 type 의 변환을 나타낸다. 묵시적인 것을 포함한 모든 변환들은 CONVERT_EXPR 혹은 NOP_EXPR node 들로 반드시 표현되어야 한다.

```
<convert_expr 0x402d2dfc type <pointer_type 0x401f1770>
    arg 0 <non_lvalue_expr 0x402d2de8
        type <integer_type 0x402520e0 size_t unsigned SI
            size <integer_cst 0x401eaf00 32>
            unit size <integer_cst 0x401eafa0 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401eafe0 0>
            max <integer_cst 0x401ed000 4294967295>
            pointer_to_this <pointer_type 0x40271c40>>
        arg 0 <var_decl 0x402d38c0 __result
            type <integer_type 0x402520e0 size_t>
            unsigned used in_system_header common regdecl SI
            file /usr/include/bits/string2.h line 921
```

```
            size <integer_cst 0x401eaf00 32>
            unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402d3620 __strcspn_c1>
            initial <integer_cst 0x402d5060 0>>>>
<non_lvalue_expr 0x40375730 type <integer_type 0x40372700 uword>
    arg 0 <component_ref 0x403732e0 type <integer_type 0x40372700 uword>
        arg 0 <indirect_ref 0x403756f4
            type <record_type 0x4036e150 dwarf_fde packed type_0 BLK
                size <integer_cst 0x401ed2c0 64>
                unit size <integer_cst 0x401ed4e0 8>
                user align 32 symtab 0 alias set -1
                attributes <tree_list 0x4037503c>
                fields <field_decl 0x40372d90 length>
                pointer_to_this <pointer_type 0x4036e230>
                chain <type_decl 0x4036e1c0>>
            arg 0 <parm_decl 0x403763f0 f>>
    arg 1 <field_decl 0x40372d90 length>>
<convert_expr 0x40375744 type <pointer_type 0x401eec40>
    arg 0 <non_lvalue_expr 0x40375730 type <integer_type 0x40372700 uword>
        arg 0 <component_ref 0x403732e0 type <integer_type 0x40372700 uword>
            arg 0 <indirect_ref 0x403756f4
                type <record_type 0x4036e150 dwarf_fde packed type_0 BLK
                    size <integer_cst 0x401ed2c0 64>
                    unit size <integer_cst 0x401ed4e0 8>
                    user align 32 symtab 0 alias set -1
                    attributes <tree_list 0x4037503c>
                    fields <field_decl 0x40372d90 length>
                    pointer_to_this <pointer_type 0x4036e230>
                    chain <type_decl 0x4036e1c0>>
                arg 0 <parm_decl 0x403763f0 f>>
        arg 1 <field_decl 0x40372d90 length>>>
```

NOP_EXPR 1 1

생성될 때 code 가 없는 것으로 요구되어지는 변환을 나타낸다.

```
 <nop_expr 0x402f1eec
    type <pointer_type 0x401eec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402979a0>>
    side-effects
    arg 0 <call_expr 0x402ddfc0
        type <pointer_type 0x401ee7e0
            type <void_type 0x401ee770 void VOID
```

```
            align 8 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x401ee7e0>>
        unsigned SI size <integer_cst 0x401ed540 32>
        unit size <integer_cst 0x401ed5a0 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402801c0>>
    side-effects
    arg 0 <addr_expr 0x402f1e88
        type <pointer_type 0x402f0b60
            type <function_type 0x402c3380 type <pointer_type 0x401ee7e0>
                DI
                size <integer_cst 0x401ed2c0 constant 64>
                unit size <integer_cst 0x401ed4e0 constant 8>
                align 64 symtab 0 alias set -1
                arg-types <tree_list 0x402c4280
                    value <pointer_type 0x402a6380
                        type <void_type 0x402a6310 void readonly VOID
                            align 8 symtab 0 alias set -1
                            pointer_to_this <pointer_type 0x402a6380>>
                        unsigned SI size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 symtab 0 alias set -1>
                    chain <tree_list 0x402c4294
                        value <integer_type 0x401da380 int SI
                            size <integer_cst 0x401eaf00 constant 32>
                            unit size <integer_cst 0x401eafa0 constant 4>
                            align 32 symtab 0 alias set -1 precision 32
                            min <integer_cst 0x401eaf60 -2147483648>
                            max <integer_cst 0x401eaf80 2147483647>
                            pointer_to_this <pointer_type 0x401f1620>>
                        chain <tree_list 0x402c42a8
                            value <void_type 0x401ee770 void>>>>
                pointer_to_this <pointer_type 0x402f0b60>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>
        arg 0 <function_decl 0x402d3460 __rawmemchr
            type <function_type 0x402c3380>
            used public in_system_header common external defer-output QI
            file /usr/include/bits/string2.h line 387
            chain <function_decl 0x402d32a0 basename>>>
    arg 1 <tree_list 0x402f1eb0
        value <nop_expr 0x402f1e9c type <pointer_type 0x402a6380>
            arg 0 <var_decl 0x402f0a80 __retval
                type <pointer_type 0x401eec40>
                unsigned used in_system_header common regdecl SI
                file /usr/include/bits/string2.h line 1137
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 context <function_decl 0x402f07e0 __strsep_1c>
                initial <indirect_ref 0x402f1cbc>>>
        chain <tree_list 0x402f1ed8
```

```
                value <nop_expr 0x402f1ec4 type <integer_type 0x401da380 int>
                    arg 0 <parm_decl 0x402f08c0 __reject
                        type <integer_type 0x401da230 char>
                        used in_system_header QI
                        file /usr/include/bits/string2.h line 1135
                        size <integer_cst 0x401eada0 8>
                        unit size <integer_cst 0x401eadc0 1>
                        align 8 context <function_decl 0x402f07e0 __strsep_1c>
                        result <integer_type 0x401da230 char>
                        initial <integer_type 0x401da380 int>
                        arg-type <integer_type 0x401da380 int>
                        arg-type-as-written <integer_type 0x401da230 char>>>>>>>
<nop_expr 0x403757e4
   type <pointer_type 0x40376380
       type <record_type 0x40376070 fde packed type_0 BLK
           size <integer_cst 0x401ed2c0 constant 64>
           unit size <integer_cst 0x401ed4e0 constant 8>
           user align 32 symtab 0 alias set -1
           attributes <tree_list 0x4037503c
               purpose <identifier_node 0x40374200 aligned>
               value <tree_list 0x40375000
                   value <integer_cst 0x403730c0 constant 4>>
               chain <tree_list 0x40375028
                   purpose <identifier_node 0x402d0fc0 packed>>>
           fields <field_decl 0x40372d90 length
               type <integer_type 0x40372700 uword unsigned SI
                   size <integer_cst 0x401eaf00 constant 32>
                   unit size <integer_cst 0x401eafa0 constant 4>
                   align 32 symtab 0 alias set -1 precision 32
                   min <integer_cst 0x401eafe0 0>
                   max <integer_cst 0x401ed000 4294967295>>
               unsigned packed SI file unwind-dw2-fde.h line 141
               size <integer_cst 0x401eaf00 32>
               unit size <integer_cst 0x401eafa0 4>
               align 8 offset_align 128
               offset <integer_cst 0x401ed5c0 constant 0>
               bit offset <integer_cst 0x401ed680 constant 0>
               context <record_type 0x4036e150 dwarf_fde>
               arguments <integer_cst 0x401ed5c0 0>
               chain <field_decl 0x40372e00 CIE_delta>>
           pointer_to_this <pointer_type 0x40376380>>
       unsigned SI
       size <integer_cst 0x401ed540 constant 32>
       unit size <integer_cst 0x401ed5a0 constant 4>
       align 32 symtab 0 alias set -1>
   arg 0 <plus_expr 0x40373420
       type <pointer_type 0x401eec40
           type <integer_type 0x401da230 char QI
               size <integer_cst 0x401eada0 constant 8>
               unit size <integer_cst 0x401eadc0 constant 1>
               align 8 symtab 0 alias set -1 precision 8
               min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
```

```
                pointer_to_this <pointer_type 0x401eec40>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x402979a0>>
        arg 0 <plus_expr 0x40373340 type <pointer_type 0x401eec40>
            arg 0 <nop_expr 0x403756e0 type <pointer_type 0x401eec40>
                arg 0 <parm_decl 0x403763f0 f type <pointer_type 0x40376380>
                    unsigned used SI file unwind-dw2-fde.h line 157
                    size <integer_cst 0x401ed540 32>
                    unit size <integer_cst 0x401ed5a0 4>
                    align 32 context <function_decl 0x403764d0 next_fde>
                    result <pointer_type 0x40376380>
                    initial <pointer_type 0x40376380>
                    arg-type <pointer_type 0x40376380>
                    arg-type-as-written <pointer_type 0x40376380>>>
            arg 1 <convert_expr 0x40375744 type <pointer_type 0x401eec40>
                arg 0 <non_lvalue_expr 0x40375730
                    type <integer_type 0x40372700 uword>
                    arg 0 <component_ref 0x403732e0
                        type <integer_type 0x40372700 uword>
                        arg 0 <indirect_ref 0x403756f4
                            type <record_type 0x4036e150 dwarf_fde packed type_0
                                BLK size <integer_cst 0x401ed2c0 64>
                                unit size <integer_cst 0x401ed4e0 8>
                                user align 32 symtab 0 alias set -1
                                attributes <tree_list 0x4037503c>
                                fields <field_decl 0x40372d90 length>
                                pointer_to_this <pointer_type 0x4036e230>
                                chain <type_decl 0x4036e1c0>>
                            arg 0 <parm_decl 0x403763f0 f>>
                        arg 1 <field_decl 0x40372d90 length>>>>>
        arg 1 <integer_cst 0x40373400 constant 4>>>
```

NON_LVALUE_EXPR 1 1

값은 argument 와 같지만, lvalue 가 아님을 보장한다.

```
<non_lvalue_expr 0x40287af0
   type <integer_type 0x401da3f0 unsigned int unsigned SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
       min <integer_cst 0x401eafe0 0>
       max <integer_cst 0x401ed000 4294967295>>
   constant
   arg 0 <integer_cst 0x402882c0
       type <integer_type 0x401da3f0 unsigned int> constant 52>>
<non_lvalue_expr 0x402b5dd4
   type <integer_type 0x401da3f0 unsigned int unsigned SI
       size <integer_cst 0x401eaf00 constant 32>
       unit size <integer_cst 0x401eafa0 constant 4>
       align 32 symtab 0 alias set -1 precision 32
```

```
            min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
            pointer_to_this <pointer_type 0x402b17e0>>
        constant
        arg 0 <integer_cst 0x402b3340
            type <integer_type 0x401da3f0 unsigned int> constant 32>>
    <non_lvalue_expr 0x402b5dd4
        type <integer_type 0x401da3f0 unsigned int unsigned SI
            size <integer_cst 0x401eaf00 constant 32>
            unit size <integer_cst 0x401eafa0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401eafe0 0>
            max <integer_cst 0x401ed000 4294967295>
            pointer_to_this <pointer_type 0x402b17e0>>
        constant
        arg 0 <integer_cst 0x402b3340
            type <integer_type 0x401da3f0 unsigned int> constant 32>>
```

VIEW_CONVERT_EXPR 1 1

두번째 type 의 것으로써 하나의 type 의 어떤것을 보여줌을 표현한다. 이것은 Ada 에서 "Unchecked Conversion" 와 거칠게는 C 에서의 관용구 *(type2 *)&X 와 상응한다. 단 하나의 operand 는 다른 type 의 것으로써 보여져야하는 값이다. 이것은 input 의 type 과 표현식의 type 이 다른 크기를 가지고 있다면 정의되지 않는다.

이 code 는 또한 실제 data 이동이 일어나지 않는 MODIFY_EXPR 의 LHS 내에서 사용되어질 수 있다. TREE_ADDRESSABLE 는 이러한 경우에 설정되어 질 것이며 GCC 는 insn 들의 생성 없이 작동을 할수 없다면 반드시 멈춰야 한다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다.

SAVE_EXPR e 3

우리가 한번 계산하고 여러번 사용할 가능성이 있는 것을 나타낼 때 사용한다. 첫번째 operand 는 그 표현식이다. 두번째는 SAVE_EXPR 가 생성된 함수 decl 이 무엇인지를 나타내고, 세번째 operand 는 RTL 인데, 표현식이 계산된 직후에는 0 이 아니다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다.

UNSAVE_EXPR e 1

UNSAVE_EXPR 에 대해, operand 0 은 unsave 할 값이다. Unsave 함으로써 우리는 한번 이상 평가되어지는 것으로부터 보호되어지는 TARGET_EXPR 들과 SAVE_EXPR 들, CALL_EXPR 들, RTL_EXPR 들과 같은 모든 _EXPR 들이 반드시 reset 되어짐으로써 이 expr 의 새로운 expand_expr 호출이 그것들을 재 평가를 불러일으키게 할수 있음을 의미한다. 이것은 우리가 다른 장소에서 tree 를 재사용하기를 원하지만 반드시 재확장 (re-expand) 이 필요한 곳에 유용할 것이다.

이 TREE node 를 위한 적절한 예제를 찾지 못하였습니다.

RTL_EXPR e 2

이 표현식이 확장될 때 내보내져야 했던 결과로써 이미 확장되어진 어떤 RTL 을 나타낸다. 첫번째 operand 는 내보내는 RTL 이다. 이것은 insn 들의 chain 의 첫번째이다. 두번째는 결과를 위한 RTL 표현식이다. RTL_EXPR 의 생성동안 만들어진 어떤 임시의 것들은 RTL_EXPR_RTL 를 제외한 RTL_EXPR 가 확장되는 즉시 재사용되어질 수 있다.

```
 <rtl_expr 0x403a8d80
    type <real_type 0x401ee9a0 double DF
        size <integer_cst 0x401ed2c0 constant 64>
        unit size <integer_cst 0x401ed4e0 constant 8>
        align 64 symtab 0 alias set -1 precision 64
        pointer_to_this <pointer_type 0x403191c0>>
    side-effects
    rtl 0 (note 13 0 15 0x403a7340 NOTE_INSN_BLOCK_BEG)
        (insn 15 13 17 (set (reg/v/f:SI 61)
                (reg/v/f:SI 59)) -1 (nil)
            (nil))
        (insn 17 15 18 (set (reg/v/f:SI 62)
                (const_int 0 [0x0])) -1 (nil)
            (nil))
        (note 18 17 19 NOTE_INSN_DELETED)
        (note 19 18 22 NOTE_INSN_DELETED)
        (insn 22 19 24 (set (mem/f:SI (reg/f:SI 56 virtual-outgoing-args)
                [0 S4 A32])
                (reg/v/f:SI 61)) -1 (nil)
            (nil))
        (insn 24 22 26 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                        (const_int 4 [0x4])) [0 S4 A32])
                (reg/v/f:SI 62)) -1 (nil)
            (nil))
        (insn 26 24 27 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                        (const_int 8 [0x8])) [0 S4 A32])
                (const_int 0 [0x0])) -1 (nil)
            (nil))
        (call_insn 27 26 29 (set (reg:DF 8 st(0))
                (call (mem:QI (symbol_ref:SI ("__strtod_internal")) [0 S1 A8])
                    (const_int 12 [0xc]))) -1 (nil)
            (nil)
            (nil))
        (insn 29 27 30 (set (reg/v:DF 63)
                (reg:DF 8 st(0))) -1 (nil)
            (nil))
        (jump_insn 30 29 31 (set (pc)
                (label_ref 33)) -1 (nil)
            (nil))
        (barrier 31 30 32)
        (note 32 31 33 0x403a7340 NOTE_INSN_BLOCK_END)
        (code_label 33 32 0 122 "" "" [0 uses])
    rtl 1 (reg/v:DF 63)
>
 <rtl_expr 0x403aa2c0
    type <integer_type 0x401da460 long int SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set 23 precision 32
        min <integer_cst 0x401ed020 -2147483648>
        max <integer_cst 0x401ed040 2147483647>
        pointer_to_this <pointer_type 0x401f17e0>>
```

```
        side-effects
        rtl 0 (note 13 0 15 0x403a7500 NOTE_INSN_BLOCK_BEG)
            (insn 15 13 17 (set (reg/v/f:SI 61)
                    (reg/v/f:SI 59)) -1 (nil)
                (nil))
            (insn 17 15 19 (set (reg/v/f:SI 62)
                    (const_int 0 [0x0])) -1 (nil)
                (nil))
            (insn 19 17 20 (set (reg/v:SI 63)
                    (const_int 10 [0xa])) -1 (nil)
                (nil))
            (note 20 19 21 NOTE_INSN_DELETED)
            (note 21 20 24 NOTE_INSN_DELETED)
            (insn 24 21 26 (set (mem/f:SI (reg/f:SI 56 virtual-outgoing-args)
                    [0 S4 A32])
                    (reg/v/f:SI 61)) -1 (nil)
                (nil))
            (insn 26 24 28 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                            (const_int 4 [0x4])) [0 S4 A32])
                    (reg/v/f:SI 62)) -1 (nil)
                (nil))
            (insn 28 26 30 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                            (const_int 8 [0x8])) [0 S4 A32])
                    (reg/v:SI 63)) -1 (nil)
                (nil))
            (insn 30 28 31 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                            (const_int 12 [0xc])) [0 S4 A32])
                    (const_int 0 [0x0])) -1 (nil)
                (nil))
            (call_insn 31 30 33 (set (reg:SI 0 eax)
                    (call (mem:QI (symbol_ref:SI ("__strtol_internal")) [0 S1 A8])
                        (const_int 16 [0x10]))) -1 (nil)
                (nil)
                (nil))
            (insn 33 31 34 (set (reg/v:SI 64)
                    (reg:SI 0 eax)) -1 (nil)
                (nil))
            (jump_insn 34 33 35 (set (pc)
                    (label_ref 37)) -1 (nil)
                (nil))
            (barrier 35 34 36)
            (note 36 35 37 0x403a7500 NOTE_INSN_BLOCK_END)
            (code_label 37 36 0 124 "" "" [0 uses])
        rtl 1 (reg/v:SI 64)
    >
    <rtl_expr 0x4039f840
        type <integer_type 0x401da460 long int SI
            size <integer_cst 0x401eaf00 constant 32>
            unit size <integer_cst 0x401eafa0 constant 4>
            align 32 symtab 0 alias set 12 precision 32
            min <integer_cst 0x401ed020 -2147483648>
            max <integer_cst 0x401ed040 2147483647>
```

```
        pointer_to_this <pointer_type 0x401f17e0>>
    side-effects
    rtl 0 (note 13 0 15 0x40397cc0 NOTE_INSN_BLOCK_BEG)
        (insn 15 13 17 (set (reg/v/f:SI 61)
                (reg/v/f:SI 59)) -1 (nil)
            (nil))
        (insn 17 15 19 (set (reg/v/f:SI 62)
                (const_int 0 [0x0])) -1 (nil)
            (nil))
        (insn 19 17 20 (set (reg/v:SI 63)
                (const_int 10 [0xa])) -1 (nil)
            (nil))
        (note 20 19 21 NOTE_INSN_DELETED)
        (note 21 20 24 NOTE_INSN_DELETED)
        (insn 24 21 26 (set (mem/f:SI (reg/f:SI 56 virtual-outgoing-args)
                [0 S4 A32])
                (reg/v/f:SI 61)) -1 (nil)
            (nil))
        (insn 26 24 28 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                        (const_int 4 [0x4])) [0 S4 A32])
                (reg/v/f:SI 62)) -1 (nil)
            (nil))
        (insn 28 26 30 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                        (const_int 8 [0x8])) [0 S4 A32])
                (reg/v:SI 63)) -1 (nil)
            (nil))
        (insn 30 28 31 (set (mem/f:SI (plus:SI (reg/f:SI 56 virtual-outgoing-args)
                        (const_int 12 [0xc])) [0 S4 A32])
                (const_int 0 [0x0])) -1 (nil)
            (nil))
        (call_insn 31 30 33 (set (reg:SI 0 eax)
                (call (mem:QI (symbol_ref:SI ("__strtol_internal")) [0 S1 A8])
                    (const_int 16 [0x10]))) -1 (nil)
            (nil)
            (nil))
        (insn 33 31 34 (set (reg/v:SI 64)
                (reg:SI 0 eax)) -1 (nil)
            (nil))
        (jump_insn 34 33 35 (set (pc)
                (label_ref 37)) -1 (nil)
            (nil))
        (barrier 35 34 36)
        (note 36 35 37 0x40397cc0 NOTE_INSN_BLOCK_END)
        (code_label 37 36 0 122 "" "" [0 uses])
    rtl 1 (reg/v:SI 64)
>
```

ADDR_EXPR e 1

C 에서의 &. 값은 어떤 operand 의 값에 존재하는 주소이다. Operand 은 아마 어떤 mode 를 가질 것이고, 결과 mode 는 Pmode 이다.

```
<addr_expr 0x402b067c
```

```
        type <pointer_type 0x402afe70
            type <function_type 0x4028ecb0 type <integer_type 0x401da380 int>
                DI
                size <integer_cst 0x401ed2c0 constant 64>
                unit size <integer_cst 0x401ed4e0 constant 8>
                align 64 symtab 0 alias set -1
                arg-types <tree_list 0x4028dac8 value <integer_type 0x401da380 int>
                    chain <tree_list 0x4028dadc
                        value <pointer_type 0x4028e150
                            type <record_type 0x4028a690 _IO_FILE type_0 BLK
                                size <integer_cst 0x40288400 constant 1184>
                                unit size <integer_cst 0x402883c0 constant 148>
                                align 32 symtab 0 alias set -1
                                fields <field_decl 0x40284850 _flags
                                    type <integer_type 0x401da380 int>
                                    in_system_header SI
                                    file /usr/include/libio.h line 262
                                    size <integer_cst 0x401eaf00 32>
                                    unit size <integer_cst 0x401eafa0 4>
                                    align 32 offset_align 128
                                    offset <integer_cst 0x401ed5c0 constant 0>
                                    bit offset <integer_cst 0x401ed680 constant 0>
                                    context <record_type 0x4026be00 _IO_FILE>
                                    arguments <integer_cst 0x401ed5c0 0>
                                    chain <field_decl 0x402848c0 _IO_read_ptr>>
                                pointer_to_this <pointer_type 0x4028e150>>
                            unsigned SI
                            size <integer_cst 0x401ed540 constant 32>
                            unit size <integer_cst 0x401ed5a0 constant 4>
                            align 32 symtab 0 alias set -1>
                        chain <tree_list 0x4028daf0
                            value <void_type 0x401ee770 void VOID
                                align 8 symtab 0 alias set -1
                                pointer_to_this <pointer_type 0x401ee7e0>>>>>
                pointer_to_this <pointer_type 0x402afe70>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>
        arg 0 <function_decl 0x4028ed20 _IO_putc type <function_type 0x4028ecb0>
            used public in_system_header common external defer-output QI
            file /usr/include/libio.h line 427
            chain <function_decl 0x4028eb60 _IO_getc>>>
    <addr_expr 0x4037976c
        type <pointer_type 0x4037a690
            type <function_type 0x403707e0 type <void_type 0x401ee770 void>
                DI
                size <integer_cst 0x401ed2c0 constant 64>
                unit size <integer_cst 0x401ed4e0 constant 8>
                align 64 symtab 0 alias set -1
                arg-types <tree_list 0x4036fd34
                    value <pointer_type 0x401ee7e0 type <void_type 0x401ee770 void>
                        unsigned SI
```

```
                    size <integer_cst 0x401ed540 constant 32>
                    unit size <integer_cst 0x401ed5a0 constant 4>
                    align 32 symtab 0 alias set 7
                    pointer_to_this <pointer_type 0x402801c0>>
                chain <tree_list 0x4036fd48
                    value <pointer_type 0x4036eee0
                        type <record_type 0x4036e460 object type_0 BLK
                            size <integer_cst 0x401ed7e0 constant 192>
                            unit size <integer_cst 0x401ed800 constant 24>
                            align 32 symtab 0 alias set -1
                            fields <field_decl 0x4036e540 pc_begin
                                type <pointer_type 0x401ee7e0>
                                unsigned SI file unwind-dw2-fde.h line 42
                                size <integer_cst 0x401ed540 32>
                                unit size <integer_cst 0x401ed5a0 4>
                                align 32 offset_align 128
                                offset <integer_cst 0x401ed5c0 constant 0>
                                bit offset <integer_cst 0x401ed680 constant 0>
                                context <record_type 0x4036e460 object>
                                arguments <integer_cst 0x401ed5c0 0>
                                chain <field_decl 0x4036e5b0 tbase>>
                            pointer_to_this <pointer_type 0x4036eee0>
                            chain <type_decl 0x4036e4d0>>
                        unsigned SI size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 symtab 0 alias set -1>
                    chain <tree_list 0x4036fd5c value <pointer_type 0x401ee7e0>
                        chain <tree_list 0x4036fd70
                            value <pointer_type 0x401ee7e0>
                            chain <tree_list 0x4036fd84
                                value <void_type 0x401ee770 void>>>>>>
            pointer_to_this <pointer_type 0x4037a690>>
        unsigned SI size <integer_cst 0x401ed540 32>
        unit size <integer_cst 0x401ed5a0 4>
        align 32 symtab 0 alias set -1>
    arg 0 <function_decl 0x40370850 __register_frame_info_bases
        type <function_type 0x403707e0>
        addressable used public common external weak defer-output QI
        file crtstuff.c line 120 attributes <tree_list 0x40375f14>
        chain <type_decl 0x40370460>>>
```

REFERENCE_EXPR e 1

Object 로의 비-lvalue reference 혹은 pointer.

C 에서 사용되지 않는 TREE node 입니다.

ENTRY_VALUE_EXPR e 1

Operand 는 함수 상수이다;; 결과는 type EPmode 의 함수 변수값이다. Static chain 이 필요한 언어에서만 사용된다.

C 에서 사용되지 않는 TREE node 입니다.

FDESC_EXPR e 2

Operand0 은 함수 상수이다; 결과는 type ptr_mode 의 함수 descriptor 의 part N 이다.

C 에서 사용되지 않는 TREE node 입니다.

COMPLEX_EXPR 2 2

주어진 같은 type 의 두 실수 혹은 정수 operand 로 대응하는 복소수 type 의 복소수 값을 반환한다.

```
<complex_expr 0x401e0600
    type <complex_type 0x401deb60 complex double
        type <real_type 0x401de9a0 double DF
            size <integer_cst 0x401d7900 constant 64>
            unit size <integer_cst 0x401d7b20 constant 8>
            align 64 symtab 0 alias set -1 precision 64>
        DC
        size <integer_cst 0x401d7de0 constant 128>
        unit size <integer_cst 0x401d7e00 constant 16>
        align 64 symtab 0 alias set -1>
    constant
    arg 0 <real_cst 0x401f5a00 type <real_type 0x401de9a0 double>
        constant 0.00000000000000000000e0>
    arg 1 <real_cst 0x401f5a40 type <real_type 0x401de9a0 double>
        constant 0.00000000000000000000e0>>
<complex_expr 0x4022f400
    type <complex_type 0x402355b0
        type <real_type 0x401de930 float SF
            size <integer_cst 0x401d7b80 constant 32>
            unit size <integer_cst 0x401d7be0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            pointer_to_this <pointer_type 0x401dfcb0>>
        SC
        size <integer_cst 0x401d7900 constant 64>
        unit size <integer_cst 0x401d7b20 constant 8>
        align 32 symtab 0 alias set -1>

    arg 0 <float_expr 0x4023499c type <real_type 0x401de930 float>

        arg 0 <var_decl 0x40235540 j
            type <integer_type 0x401da380 int SI
                size <integer_cst 0x401d7540 constant 32>
                unit size <integer_cst 0x401d75e0 constant 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401d75a0 -2147483648>
                max <integer_cst 0x401d75c0 2147483647>
                pointer_to_this <pointer_type 0x401e2620>>
            used common SI file <stdin> line 5 size <integer_cst 0x401d7540 32>
            unit size <integer_cst 0x401d75e0 4>
            align 32 context <function_decl 0x402353f0 zz>
            initial <integer_cst 0x4022f3c0 3>>>
    arg 1 <real_cst 0x40236000 type <real_type 0x401de930 float>
        constant 0.00000000000000000000e0>>
```

CONJ_EXPR 1 1

Operand 의 복소수 켤레 (비슷한 말로는 공액, 공역). 오직 복소수 type 에서만 사용됨.

이 TREE node 를 위한 적당한 예제를 찾지 못하였습니다. 이에 대한 적절한 예제를 가지고 계신 분은 저에게 보내주시면 감사하겠습니다.

REALPART_EXPR 1 1

오직 복소수 type 의 operand 상에서만 사용되며, 그것들은 대응하는 component type 의 값을 반환한다.

```
<realpart_expr 0x401e5514
    type <real_type 0x401de930 float SF
        size <integer_cst 0x401d7b80 constant 32>
        unit size <integer_cst 0x401d7be0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        pointer_to_this <pointer_type 0x401dfcb0>>
    arg 0 <parm_decl 0x401f6540 c
        type <complex_type 0x401deaf0 complex float
            type <real_type 0x401de930 float>
            SC
            size <integer_cst 0x401d7900 constant 64>
            unit size <integer_cst 0x401d7b20 constant 8>
            align 32 symtab 0 alias set -1>
        used SC file <stdin> line 2 size <integer_cst 0x401d7900 64>
        unit size <integer_cst 0x401d7b20 8>
        align 32 context <function_decl 0x401f6620 realpart>
        result <complex_type 0x401deaf0 complex float>
        initial <complex_type 0x401deaf0 complex float>
        arg-type <complex_type 0x401deaf0 complex float>
        arg-type-as-written <complex_type 0x401deaf0 complex float>>>
<realpart_expr 0x401e5514
    type <integer_type 0x401da460 long int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d7660 -2147483648>
        max <integer_cst 0x401d7680 2147483647>
        pointer_to_this <pointer_type 0x401e27e0>>
    arg 0 <parm_decl 0x401f65b0 c
        type <complex_type 0x401f6540 type <integer_type 0x401da460 long int>
            CSI
            size <integer_cst 0x401d7900 constant 64>
            unit size <integer_cst 0x401d7b20 constant 8>
            align 32 symtab 0 alias set -1>
        used CSI file <stdin> line 2 size <integer_cst 0x401d7900 64>
        unit size <integer_cst 0x401d7b20 8>
        align 32 context <function_decl 0x401f6690 realpart>
        result <complex_type 0x401f6540> initial <complex_type 0x401f6540>
        arg-type <complex_type 0x401f6540>
        arg-type-as-written <complex_type 0x401f6540>>>
<realpart_expr 0x40234ce4
```

```
    type <real_type 0x401de930 float SF
        size <integer_cst 0x401d7b80 constant 32>
        unit size <integer_cst 0x401d7be0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        pointer_to_this <pointer_type 0x401dfcb0>>
    arg 0 <var_decl 0x402359a0 a
        type <complex_type 0x401deaf0 complex float
            type <real_type 0x401de930 float>
            SC
            size <integer_cst 0x401d7900 constant 64>
            unit size <integer_cst 0x401d7b20 constant 8>
            align 32 symtab 0 alias set -1>
        used common SC file <stdin> line 14 size <integer_cst 0x401d7900 64>
        unit size <integer_cst 0x401d7b20 8>
        align 32 context <function_decl 0x40235690 main>>>
```

IMAGPART_EXPR 1 1

오직 복소수 type 의 operand 상에서만 사용되며, 그것들은 대응하는 component type 의 값을 반환한다.

```
<imagpart_expr 0x401e55a0
    type <real_type 0x401de930 float SF
        size <integer_cst 0x401d7b80 constant 32>
        unit size <integer_cst 0x401d7be0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        pointer_to_this <pointer_type 0x401dfcb0>>
    arg 0 <parm_decl 0x401f6540 c
        type <complex_type 0x401deaf0 complex float
            type <real_type 0x401de930 float>
            SC
            size <integer_cst 0x401d7900 constant 64>
            unit size <integer_cst 0x401d7b20 constant 8>
            align 32 symtab 0 alias set -1>
        used SC file <stdin> line 2 size <integer_cst 0x401d7900 64>
        unit size <integer_cst 0x401d7b20 8>
        align 32 context <function_decl 0x401f6690 realpart>
        result <complex_type 0x401deaf0 complex float>
        initial <complex_type 0x401deaf0 complex float>
        arg-type <complex_type 0x401deaf0 complex float>
        arg-type-as-written <complex_type 0x401deaf0 complex float>
        chain <parm_decl 0x401f65b0 a
            type <integer_type 0x401da380 int SI
                size <integer_cst 0x401d7540 constant 32>
                unit size <integer_cst 0x401d75e0 constant 4>
                align 32 symtab 0 alias set -1 precision 32
                min <integer_cst 0x401d75a0 -2147483648>
                max <integer_cst 0x401d75c0 2147483647>
                pointer_to_this <pointer_type 0x401e2620>>
            SI file <stdin> line 2 size <integer_cst 0x401d7540 32>
            unit size <integer_cst 0x401d75e0 4>
            align 32 context <function_decl 0x401f6690 realpart>
            result <integer_type 0x401da380 int>
```

```
            initial <integer_type 0x401da380 int>
            arg-type <integer_type 0x401da380 int>
            arg-type-as-written <integer_type 0x401da380 int>>>>
    <imagpart_expr 0x401e5514
        type <integer_type 0x401da460 long int SI
            size <integer_cst 0x401d7540 constant 32>
            unit size <integer_cst 0x401d75e0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            min <integer_cst 0x401d7660 -2147483648>
            max <integer_cst 0x401d7680 2147483647>
            pointer_to_this <pointer_type 0x401e27e0>>
        arg 0 <parm_decl 0x401f65b0 c
            type <complex_type 0x401f6540 type <integer_type 0x401da460 long int>
                CSI
                size <integer_cst 0x401d7900 constant 64>
                unit size <integer_cst 0x401d7b20 constant 8>
                align 32 symtab 0 alias set -1>
            used CSI file <stdin> line 2 size <integer_cst 0x401d7900 64>
            unit size <integer_cst 0x401d7b20 8>
            align 32 context <function_decl 0x401f6690 realpart>
            result <complex_type 0x401f6540> initial <complex_type 0x401f6540>
            arg-type <complex_type 0x401f6540>
            arg-type-as-written <complex_type 0x401f6540>>>
    <imagpart_expr 0x40234d0c
        type <real_type 0x401de930 float SF
            size <integer_cst 0x401d7b80 constant 32>
            unit size <integer_cst 0x401d7be0 constant 4>
            align 32 symtab 0 alias set -1 precision 32
            pointer_to_this <pointer_type 0x401dfcb0>>
        arg 0 <var_decl 0x402359a0 a
            type <complex_type 0x401deaf0 complex float
                type <real_type 0x401de930 float>
                SC
                size <integer_cst 0x401d7900 constant 64>
                unit size <integer_cst 0x401d7b20 constant 8>
                align 32 symtab 0 alias set -1>
            used common SC file <stdin> line 14 size <integer_cst 0x401d7900 64>
            unit size <integer_cst 0x401d7b20 8>
            align 32 context <function_decl 0x40235690 main>>>
```

PREDECREMENT_EXPR e 2

C 에서 ++ 과 – 를 위한 node 들.
두번째 arg 는 얼마나 많이 증가할꺼냐 혹은 감사할꺼냐를 가르킨다. 포인터에 대해서는 가르켜진 object 의 크기일 것이다.

```
 <preincrement_expr 0x402f2500
    type <pointer_type 0x401eec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
```

```
            pointer_to_this <pointer_type 0x401eec40>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402979a0>>
    side-effects
    arg 0 <var_decl 0x402f41c0 __cp type <pointer_type 0x401eec40>
        unsigned used in_system_header common regdecl SI
        file /usr/include/bits/string2.h line 1150
        size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
        align 32 context <function_decl 0x402f0e00 __strsep_2c>
        initial <var_decl 0x402f4150 __retval>>
    arg 1 <integer_cst 0x402f24e0 type <pointer_type 0x401eec40> constant 1>>
<preincrement_expr 0x402f2920
    type <pointer_type 0x401eec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401eec40>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x402979a0>>
    side-effects
    arg 0 <var_decl 0x402f4850 __cp type <pointer_type 0x401eec40>
        unsigned used in_system_header common regdecl SI
        file /usr/include/bits/string2.h line 1178
        size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
        align 32 context <function_decl 0x402f4460 __strsep_3c>
        initial <var_decl 0x402f47e0 __retval>>
    arg 1 <integer_cst 0x402f2900 type <pointer_type 0x401eec40> constant 1>>
<preincrement_expr 0x402dd620
    type <pointer_type 0x401f1770
        type <integer_type 0x401f1700 char readonly QI
            size <integer_cst 0x401eada0 constant 8>
            unit size <integer_cst 0x401eadc0 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401eae60 -128> max <integer_cst 0x401eae80 127>
            pointer_to_this <pointer_type 0x401f1770>>
        unsigned SI
        size <integer_cst 0x401ed540 constant 32>
        unit size <integer_cst 0x401ed5a0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x4027ee00>>
    side-effects
    arg 0 <parm_decl 0x402dcc40 __s type <pointer_type 0x401f1770>
        unsigned used in_system_header SI
        file /usr/include/bits/string2.h line 1045 size <integer_cst 0x401ed540 32>
```

```
            unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402dcbd0 __strpbrk_c3>
            result <pointer_type 0x401f1770> initial <pointer_type 0x401f1770>
            arg-type <pointer_type 0x401f1770>
            arg-type-as-written <pointer_type 0x401f1770>
            chain <parm_decl 0x402dccb0 __accept1
                type <integer_type 0x401da380 int SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eaf60 -2147483648>
                    max <integer_cst 0x401eaf80 2147483647>
                    pointer_to_this <pointer_type 0x401f1620>>
                used in_system_header SI file /usr/include/bits/string2.h line 1045
                size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
                align 32 context <function_decl 0x402dcbd0 __strpbrk_c3>
                result <integer_type 0x401da380 int>
                initial <integer_type 0x401da380 int>
                arg-type <integer_type 0x401da380 int>
                arg-type-as-written <integer_type 0x401da380 int>
                chain <parm_decl 0x402dcd20 __accept2>>>
        arg 1 <integer_cst 0x402dd600 type <pointer_type 0x401f1770> constant 1>>
```

PREINCREMENT_EXPR e 2

C 에서 ++ 과 – 를 위한 node 들.
두번째 arg 는 얼마나 많이 증가할꺼냐 혹은 감사할꺼냐를 가르킨다. 포인터에 대해서는 가르켜진 object 의 크기일 것이다.

```
<preincrement_expr 0x402d5280
    type <integer_type 0x402520e0 size_t unsigned SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
        pointer_to_this <pointer_type 0x40271c40>>
    side-effects
    arg 0 <var_decl 0x402d38c0 __result type <integer_type 0x402520e0 size_t>
        unsigned used in_system_header common regdecl SI
        file /usr/include/bits/string2.h line 921 size <integer_cst 0x401eaf00 32>
        unit size <integer_cst 0x401eafa0 4>
        align 32 context <function_decl 0x402d3620 __strcspn_c1>
        initial <integer_cst 0x402d5060 0>>
    arg 1 <integer_cst 0x402d5260
        type <integer_type 0x402520e0 size_t> constant 1>>
<preincrement_expr 0x402dd040
    type <integer_type 0x402520e0 size_t unsigned SI
        size <integer_cst 0x401eaf00 constant 32>
        unit size <integer_cst 0x401eafa0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401eafe0 0> max <integer_cst 0x401ed000 4294967295>
        pointer_to_this <pointer_type 0x40271c40>>
    side-effects
```

```
        arg 0 <var_decl 0x402dc460 __result type <integer_type 0x402520e0 size_t>
            unsigned used in_system_header common regdecl SI
            file /usr/include/bits/string2.h line 1002 size <integer_cst 0x401eaf00 32>
            unit size <integer_cst 0x401eafa0 4>
            align 32 context <function_decl 0x402dc0e0 __strspn_c3>
            initial <integer_cst 0x402d5e20 0>>
        arg 1 <integer_cst 0x402dd020
            type <integer_type 0x402520e0 size_t> constant 1>>
    <preincrement_expr 0x402dd260
        type <pointer_type 0x401f1770
            type <integer_type 0x401f1700 char readonly QI
                size <integer_cst 0x401eada0 constant 8>
                unit size <integer_cst 0x401eadc0 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401eae60 -128>
                max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401f1770>>
            unsigned SI
            size <integer_cst 0x401ed540 constant 32>
            unit size <integer_cst 0x401ed5a0 constant 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x4027ee00>>
        side-effects
        arg 0 <parm_decl 0x402dc700 __s type <pointer_type 0x401f1770>
            unsigned used in_system_header SI
            file /usr/include/bits/string2.h line 1034
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402dc690 __strpbrk_c2>
            result <pointer_type 0x401f1770> initial <pointer_type 0x401f1770>
            arg-type <pointer_type 0x401f1770>
            arg-type-as-written <pointer_type 0x401f1770>
            chain <parm_decl 0x402dc770 __accept1
                type <integer_type 0x401da380 int SI
                    size <integer_cst 0x401eaf00 constant 32>
                    unit size <integer_cst 0x401eafa0 constant 4>
                    align 32 symtab 0 alias set -1 precision 32
                    min <integer_cst 0x401eaf60 -2147483648>
                    max <integer_cst 0x401eaf80 2147483647>
                    pointer_to_this <pointer_type 0x401f1620>>
                used in_system_header SI file /usr/include/bits/string2.h line 1034
                size <integer_cst 0x401eaf00 32> unit size <integer_cst 0x401eafa0 4>
                align 32 context <function_decl 0x402dc690 __strpbrk_c2>
                result <integer_type 0x401da380 int>
                initial <integer_type 0x401da380 int>
                arg-type <integer_type 0x401da380 int>
                arg-type-as-written <integer_type 0x401da380 int>
                chain <parm_decl 0x402dc7e0 __accept2>>>
        arg 1 <integer_cst 0x402dd240 type <pointer_type 0x401f1770> constant 1>>
```

POSTDECREMENT_EXPR e 2

C 에서 ++ 과 – 를 위한 node 들.
두번째 arg 는 얼마나 많이 증가할꺼냐 혹은 감사할꺼냐를 가르킨다. 포인터에 대해서는 가르

켜진 object 의 크기일 것이다.

```
<postdecrement_expr 0x40373bc0
   type <pointer_type 0x40378380
       type <pointer_type 0x403781c0 func_ptr
           type <function_type 0x401f1af0
               type <void_type 0x401ee770 void VOID
                   align 8 symtab 0 alias set -1
                   pointer_to_this <pointer_type 0x401ee7e0>>
               DI
               size <integer_cst 0x401ed2c0 constant 64>
               unit size <integer_cst 0x401ed4e0 constant 8>
               align 64 symtab 0 alias set -1
               arg-types <tree_list 0x401d8974 value <void_type 0x401ee770 void>>
               pointer_to_this <pointer_type 0x4031e3f0>>
           unsigned SI
           size <integer_cst 0x401ed540 constant 32>
           unit size <integer_cst 0x401ed5a0 constant 4>
           align 32 symtab 0 alias set -1
           pointer_to_this <pointer_type 0x40378380>>
       unsigned SI size <integer_cst 0x401ed540 32>
       unit size <integer_cst 0x401ed5a0 4>
       align 32 symtab 0 alias set -1>
   side-effects
   arg 0 <var_decl 0x40378cb0 p type <pointer_type 0x40378380>
       unsigned used common SI file crtstuff.c line 482
       size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
       align 32 context <function_decl 0x40378bd0 __do_global_ctors_aux>>
   arg 1 <integer_cst 0x40373ba0 type <pointer_type 0x40378380> constant 4>>
```

POSTINCREMENT_EXPR e 2

C 에서 ++ 과 – 를 위한 node 들.
두번째 arg 는 얼마나 많이 증가할꺼냐 혹은 감사할꺼냐를 가르킨다. 포인터에 대해서는 가르
켜진 object 의 크기일 것이다.

```
<postincrement_expr 0x40373d20
   type <pointer_type 0x40378380
       type <pointer_type 0x403781c0 func_ptr
           type <function_type 0x401f1af0
               type <void_type 0x401ee770 void VOID
                   align 8 symtab 0 alias set -1
                   pointer_to_this <pointer_type 0x401ee7e0>>
               DI
               size <integer_cst 0x401ed2c0 constant 64>
               unit size <integer_cst 0x401ed4e0 constant 8>
               align 64 symtab 0 alias set -1
               arg-types <tree_list 0x401d8974 value <void_type 0x401ee770 void>>
               pointer_to_this <pointer_type 0x4031e3f0>>
           unsigned SI
           size <integer_cst 0x401ed540 constant 32>
           unit size <integer_cst 0x401ed5a0 constant 4>
```

```
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x40378380>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>
        side-effects
        arg 0 <var_decl 0x40378ee0 p type <pointer_type 0x40378380>
            unsigned used static common SI file crtstuff.c line 255
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x40378e00 __do_global_dtors_aux>
            initial <plus_expr 0x40373bc0>>
        arg 1 <integer_cst 0x40373d00 type <pointer_type 0x40378380> constant 4>>
    <postincrement_expr 0x402a0ac0 type <pointer_type 0x401eec40>
        side-effects
        arg 0 <component_ref 0x402a0a80 type <pointer_type 0x401eec40>
            arg 0 <indirect_ref 0x402b0424 type <record_type 0x4026be00 _IO_FILE>
                arg 0 <var_decl 0x402929a0 stdin>>
            arg 1 <field_decl 0x402848c0 _IO_read_ptr>>
        arg 1 <integer_cst 0x402a0aa0 constant 1>>
    <postincrement_expr 0x40373da0
        type <pointer_type 0x40378380
            type <pointer_type 0x403781c0 func_ptr
                type <function_type 0x401f1af0
                    type <void_type 0x401ee770 void VOID
                        align 8 symtab 0 alias set -1
                        pointer_to_this <pointer_type 0x401ee7e0>>
                    DI
                    size <integer_cst 0x401ed2c0 constant 64>
                    unit size <integer_cst 0x401ed4e0 constant 8>
                    align 64 symtab 0 alias set -1
                    arg-types <tree_list 0x401d8974 value <void_type 0x401ee770 void>>
                    pointer_to_this <pointer_type 0x4031e3f0>>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1
                pointer_to_this <pointer_type 0x40378380>>
            unsigned SI size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 symtab 0 alias set -1>
        side-effects
        arg 0 <var_decl 0x40378ee0 p type <pointer_type 0x40378380>
            unsigned used static common SI file crtstuff.c line 255
            size <integer_cst 0x401ed540 32> unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x40378e00 __do_global_dtors_aux>
            initial <plus_expr 0x40373bc0>>
        arg 1 <integer_cst 0x40373d80 type <pointer_type 0x40378380> constant 4>>
```

VA_ARG_EXPR e 1

'va_arg' 를 수행하는데 사용된다.

```
<va_arg_expr 0x40260064
```

```
    type <integer_type 0x401da380 int SI
        size <integer_cst 0x401d7540 constant 32>
        unit size <integer_cst 0x401d75e0 constant 4>
        align 32 symtab 0 alias set -1 precision 32
        min <integer_cst 0x401d75a0 -2147483648>
        max <integer_cst 0x401d75c0 2147483647>
        pointer_to_this <pointer_type 0x401e2620>>
    side-effects
    arg 0 <var_decl 0x4025ca10 argument
        type <pointer_type 0x402353f0 va_list
            type <integer_type 0x401da230 char QI
                size <integer_cst 0x401d73e0 constant 8>
                unit size <integer_cst 0x401d7400 constant 1>
                align 8 symtab 0 alias set -1 precision 8
                min <integer_cst 0x401d74a0 -128> max <integer_cst 0x401d74c0 127>
                pointer_to_this <pointer_type 0x401dec40>>
            unsigned SI
            size <integer_cst 0x401d7b80 constant 32>
            unit size <integer_cst 0x401d7be0 constant 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x4025cd20>>
        addressable unsigned used common SI file <stdin> line 24
        size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
        align 32 context <function_decl 0x40259bd0 test_fn>>>
<va_arg_expr 0x402602a8
    type <pointer_type 0x401dec40
        type <integer_type 0x401da230 char QI
            size <integer_cst 0x401d73e0 constant 8>
            unit size <integer_cst 0x401d7400 constant 1>
            align 8 symtab 0 alias set -1 precision 8
            min <integer_cst 0x401d74a0 -128> max <integer_cst 0x401d74c0 127>
            pointer_to_this <pointer_type 0x401dec40>>
        unsigned SI
        size <integer_cst 0x401d7b80 constant 32>
        unit size <integer_cst 0x401d7be0 constant 4>
        align 32 symtab 0 alias set -1
        pointer_to_this <pointer_type 0x40235f50>>
    side-effects
    arg 0 <var_decl 0x4025ca10 argument
        type <pointer_type 0x402353f0 va_list type <integer_type 0x401da230 char>
            unsigned SI size <integer_cst 0x401d7b80 32>
            unit size <integer_cst 0x401d7be0 4>
            align 32 symtab 0 alias set -1
            pointer_to_this <pointer_type 0x4025cd20>>
        addressable unsigned used common SI file <stdin> line 24
        size <integer_cst 0x401d7b80 32> unit size <integer_cst 0x401d7be0 4>
        align 32 context <function_decl 0x40259bd0 test_fn>>>
```

TRY_CATCH_EXPR e 2

Operand 1 을 평가한다. operand 1 의 평가가 이루어지는 동안 예외상황이 던져질 경우, operand 2 를 평가한다.

이것은 예외상황이 던져지지 않으면 operand 2 가 절대 평가되지 않는다는 점에서 WITH_CLEANUP_EXPR 와는 다르다.

C 에서 사용되지 않는 TREE node 입니다.

TRY_FINALLY_EXPR e 2

첫번째 operand 를 평가한다. 두번째 operand 는 이 표현식으로부터 exit (normal, 혹은 exception, jump out) 하기 전에 평가되어진 cleanup 표현식이다.

CLEANUP_POINT_EXPR/WITH_CLEANUP_EXPR 조합과 비슷하지만, 그것들은 항상 필요한 곳에 cleanup 표현식을 복사한다. 반대로, TRY_FINALLY_EXPR 는 cleanup 하위루틴으로의 jump 를 생성한다. (적어도 개념상으로; Optimizer 는 보통의 하위루틴을 inline 할수있는 것과 같은 방식으로 cleanup 하위루틴을 inline 할 수 있다.) TRY_FINALLY_EXPR 는 cleanup 이 (사람들이 breakpoint 를 지정하기를 원하는) 현재 함수의 source 내부의 실제 statement 일 경우 반드시 사용되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

GOTO_SUBROUTINE_EXPR e 2

TRY_FINALLY_EXPR 의 기능에서 cleanup 들을 위해 내부적으로 사용된다. (특히, front-end 들에서 말고, expand_expr 에 의해 생성된다.) Operand 0 은 우리가 호출할 필요가 있는 하위루틴의 시작점을 위한 rtx 이다. Operand 1 은 하위루틴이 반드시 어디서 반환되어야 하는지에 대한 주소를 저장하는 변수를 위한 rtx 이다.

C 에서 사용되지 않는 TREE node 입니다.

이 표현식의 type 들은 유용한 값들을 가지고 있지 않고 항상 부가적 작용을 가지고 있다.

LABEL_EXPR s 1

Statement 로써 캡슐로 싸여진 라벨 정의. Operand 0 은 여기서 보인 라벨을 위한 LABEL_DECL node 이다. Type 은 반드시 void 여야 하고 값은 무시되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

GOTO_EXPR s 1

GOTO. Operand 0 은 LABEL_DECL node 혹은 표현식이다. Type 은 반드시 void 여야 하고 값은 무시되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

RETURN_EXPR s 1

RETURN. operand 0, 을 평가한 후 현재 현수로부터 돌아가야 한다. 추측하건데 그 operand 는 반환된 값을 가지고 있는 RESULT_DECL 내에 저장된 assignment 이다. Operand 는 아마 null 일 것이다.
Type 은 반드시 void 여야 하고 값은 무시되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

EXIT_EXPR s 1

조건에 따라 가장 내부 loop 를 빠져나온다. Operand 0 은 조건이다. Type 은 반드시 void 여야 하고 값은 무시되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

LOOP_EXPR s 1

Loop. Operand 0 은 loop 의 body 이다.
이것은 반드시 EXIT_EXPR 를 포함하였거나 혹은 영구 loop 이다.
Type 은 반드시 void 여야 하고 값은 무시되어야 한다.

C 에서 사용되지 않는 TREE node 입니다.

LABELED_BLOCK_EXPR e 2

라벨을 단 block. Operand 0 은 block 의 끝으로 mark 된 것으로 생성된 라벨이다.
Operand 1 은 라벨을 단 block body 이다.

C 에서 사용되지 않는 TREE node 입니다.

EXIT_BLOCK_EXPR e 2

라벨을 단 block 을 빠져나오며, 가능한 값을 반환한다. Operand 0 은 빠져나올 LABELED_BLOCK_EXPR 이다. Operand 1 은 반환할 값이다. 이것은 NULL 로 남겨질 수 있다.

C 에서 사용되지 않는 TREE node 입니다.

EXPR_WITH_FILE_LOCATION e 3

Source 위치 정보: 파일 이름 (EXPR_WFL_FILENAME); 행 번호 ((EXPR_WFL_LINENO); 열 번호 (EXPR_WFL_COLNO) 를 가진 tree node (보통의 표현식) 에 주석을 단다. 이것은 contained node (EXPR_WFL_NODE); 로써 확장되어진다; 만약 EXPR_WFL_EMIT_LINE_NOTE 라면 line note 가 처음 내보내져야 한다. 세번째 operand 는 오직 Java front-end 에서만 사용되며, 나중에는 제거될 것이다.

```
<expr_with_file_location 0x403a8c40
   type <real_type 0x401ee9a0 double DF
       size <integer_cst 0x401ed2c0 constant 64>
       unit size <integer_cst 0x401ed4e0 constant 8>
       align 64 symtab 0 alias set -1 precision 64
       pointer_to_this <pointer_type 0x403191c0>>
   side-effects used public
   arg 0 <stmt_expr 0x40397424 type <real_type 0x401ee9a0 double>
       side-effects tree_0
       arg 0 <scope_stmt 0x40397460 tree_0 tree_3
           arg 0 <block 0x403a7340 used
               vars <var_decl 0x4037abd0 __nptr
                   type <pointer_type 0x40290690
                       type <integer_type 0x401f1700 char readonly QI
                           size <integer_cst 0x401eada0 constant 8>
                           unit size <integer_cst 0x401eadc0 constant 1>
                           align 8 symtab 0 alias set -1 precision 8
```

```
                min <integer_cst 0x401eae60 -128>
                max <integer_cst 0x401eae80 127>
                pointer_to_this <pointer_type 0x401f1770>>
            unsigned SI
            size <integer_cst 0x401ed540 constant 32>
            unit size <integer_cst 0x401ed5a0 constant 4>
            align 32 symtab 0 alias set -1>
        unsigned used SI file /usr/include/stdlib.h line 295
        size <integer_cst 0x401ed540 32>
        unit size <integer_cst 0x401ed5a0 4>
        align 32 context <function_decl 0x402fc3f0 atof>
        abstract_origin <parm_decl 0x4030bd90 __nptr>
        initial <parm_decl 0x40310770 __nptr>
        chain <var_decl 0x4037ac40 __endptr>>
    abstract_origin <function_decl 0x402fc9a0 strtod
        type <function_type 0x402fc930
            TYPE <real_type 0x401ee9a0 double>
            DI size <integer_cst 0x401ed2c0 64>
            unit size <integer_cst 0x401ed4e0 8>
            align 64 symtab 0 alias set -1
            arg-types <tree_list 0x402fd3fc
                value <pointer_type 0x40290690>
                chain <tree_list 0x402fd410
                    value <pointer_type 0x40297a10>
                    chain <tree_list 0x402fd424>>>
            pointer_to_this <pointer_type 0x40310930>>
        used nothrow public in_system_header external inline
        defer-output QI file /usr/include/stdlib.h line 295

        arguments <parm_decl 0x4030bd90 __nptr
        type <pointer_type 0x40290690>
            unsigned used in_system_header SI
            file /usr/include/stdlib.h line 294
            size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 alias set -2
            context <function_decl 0x402fc9a0 strtod>
            result <pointer_type 0x40290690>
            initial <pointer_type 0x40290690>
            (reg/v/f:SI 59)
            arg-type <pointer_type 0x40290690>
            arg-type-as-written <pointer_type 0x40290690>
            incoming-rtl (mem/f:SI (reg/f:SI 53
                virtual-incoming-args) [19 __nptr+0 S4 A32])
                chain <parm_decl 0x4030be00 __endptr>>
        result <result_decl 0x4030bf50
            type <real_type 0x401ee9a0 double>
            in_system_header regdecl DF
            file /usr/include/stdlib.h line 295
            size <integer_cst 0x401ed2c0 64>
            unit size <integer_cst 0x401ed4e0 8>
            align 64 context <function_decl 0x402fc9a0 strtod>
```

```
                        (reg:DF 58)> initial <block 0x402f9400>
                    (mem:QI (symbol_ref:SI ("strtod")) [0 S1 A8])
            chain <decl_stmt 0x40397438 arg 0 <var_decl 0x4037abd0 __nptr>
                chain <decl_stmt 0x4039744c
                    arg 0 <var_decl 0x4037ac40 __endptr
                        type <pointer_type 0x40297a10>
                        unsigned used SI file /usr/include/stdlib.h line 295
                        size <integer_cst 0x401ed540 32>
                        unit size <integer_cst 0x401ed5a0 4>
                        align 32 context <function_decl 0x402fc3f0 atof>
                        abstract_origin <parm_decl 0x4030be00 __endptr>
                        initial <integer_cst 0x4030f320 0>>
                    chain <decl_stmt 0x40397488
                        arg 0 <var_decl 0x4037ad20
                            type <real_type 0x401ee9a0 double>
                            used DF file /usr/include/stdlib.h line 295
                            size <integer_cst 0x401ed2c0 64>
                            unit size <integer_cst 0x401ed4e0 8>
                            align 64 context <function_decl 0x402fc3f0 atof>
                            abstract_origin <result_decl 0x4030bf50>>
                        chain <compound_stmt 0x4039749c tree_2
                            arg 0 <scope_stmt 0x403974b0 tree_0
                                chain <expr_stmt 0x403974d8 tree_1
                                    arg 0 <modify_expr 0x403a8be0
                                        type <real_type 0x401ee9a0 double>
                                        side-effects arg 0 <var_decl 0x4037ad20>
                                        arg 1 <call_expr 0x403a8c00
                                            type <real_type 0x401ee9a0 double>
                                            side-effects
                                            arg 0 <addr_expr 0x403974ec>
                                            arg 1 <tree_list 0x40397500>>>
                                    chain <goto_stmt 0x4039753c tree_0
                                        arg 0 <label_decl 0x4037acb0 VOID>
                                        chain <scope_stmt 0x40397550>>>>
                            chain <scope_stmt 0x40397564 tree_3
                                arg 0 <block 0x403a7340>
                                chain <label_stmt 0x40397578
                                    arg 0 <label_decl 0x4037acb0>
                                    chain <expr_stmt 0x40397474 addressable
                                    arg 0 <var_decl 0x4037ad20>>>>>>>>>>
    arg 1 <identifier_node 0x403a7380 /usr/include/stdlib.h>
    /usr/include/stdlib.h:295:0>
<expr_with_file_location 0x4039fcc0
    type <integer_type 0x401da540 long long int DI
        size <integer_cst 0x401eafc0 constant 64>
        unit size <integer_cst 0x401ed0e0 constant 8>
        align 64 symtab 0 alias set 17 precision 64
        min <integer_cst 0x401ed0a0 0x8000000000000000>
        max <integer_cst 0x401ed0c0 0x7fffffffffffffff>>
    side-effects used public
    arg 0 <stmt_expr 0x4037fe10 type <integer_type 0x401da540 long long int>
        side-effects tree_0
```

```
arg 0 <scope_stmt 0x4037fe60 tree_0 tree_3
    arg 0 <block 0x40397e40 used
        vars <var_decl 0x4039b850 __nptr
            type <pointer_type 0x40290690
                type <integer_type 0x401f1700 char readonly QI
                    size <integer_cst 0x401eada0 constant 8>
                    unit size <integer_cst 0x401eadc0 constant 1>
                    align 8 symtab 0 alias set -1 precision 8
                    min <integer_cst 0x401eae60 -128>
                    max <integer_cst 0x401eae80 127>
                    pointer_to_this <pointer_type 0x401f1770>>
                unsigned SI
                size <integer_cst 0x401ed540 constant 32>
                unit size <integer_cst 0x401ed5a0 constant 4>
                align 32 symtab 0 alias set -1>
            unsigned used SI file /usr/include/stdlib.h line 343
            size <integer_cst 0x401ed540 32>
            unit size <integer_cst 0x401ed5a0 4>
            align 32 context <function_decl 0x402fc7e0 atoll>
            abstract_origin <parm_decl 0x40310230 __nptr>
            initial <parm_decl 0x40310d90 __nptr>
            chain <var_decl 0x4039b8c0 __endptr>>
        abstract_origin <function_decl 0x402fe850 strtoll
            type <function_type 0x402fe380>
            used nothrow public in_system_header external inline
            defer-output QI file /usr/include/stdlib.h line 343
            arguments <parm_decl 0x40310230 __nptr
                type <pointer_type 0x40290690>
                unsigned used in_system_header SI
                file /usr/include/stdlib.h line 341
                size <integer_cst 0x401ed540 32>
                unit size <integer_cst 0x401ed5a0 4>
                align 32 alias set -2
                context <function_decl 0x402fe850 strtoll>
                result <pointer_type 0x40290690>
                initial <pointer_type 0x40290690>
                (reg/v/f:SI 59) arg-type <pointer_type 0x40290690>
                arg-type-as-written <pointer_type 0x40290690>
                incoming-rtl (mem/f:SI (reg/f:SI 53 virtual-incoming-args)
                    [7 __nptr+0 S4 A32])
                chain <parm_decl 0x403102a0 __endptr>>
            result <result_decl 0x40310460
                type <integer_type 0x401da540 long long int>
                in_system_header regdecl DI
                file /usr/include/stdlib.h line 343
                size <integer_cst 0x401eafc0 64>
                unit size <integer_cst 0x401ed0e0 8>
                align 64 context <function_decl 0x402fe850 strtoll>
                (reg:DI 58)> initial <block 0x402f9640>
            (mem:QI (symbol_ref:SI ("strtoll")) [0 S1 A8])
            saved-insns 0x40398700
                chain <function_decl 0x402fe620 strtouq>>>
```

```
                chain <decl_stmt 0x4037fe24 arg 0 <var_decl 0x4039b850 __nptr>
                    chain <decl_stmt 0x4037fe38
                        arg 0 <var_decl 0x4039b8c0 __endptr
                            type <pointer_type 0x40297a10>
                            unsigned used SI file /usr/include/stdlib.h line 343
                            size <integer_cst 0x401ed540 32>
                            unit size <integer_cst 0x401ed5a0 4>
                            align 32 context <function_decl 0x402fc7e0 atoll>
                            abstract_origin <parm_decl 0x403102a0 __endptr>
                            initial <integer_cst 0x4030f5a0 0>
                            chain <var_decl 0x4039b930 __base>>
                        chain <decl_stmt 0x4037fe4c
                            arg 0 <var_decl 0x4039b930 __base
                                type <integer_type 0x401da380 int>
                                used SI file /usr/include/stdlib.h line 343
                                size <integer_cst 0x401eaf00 32>
                                unit size <integer_cst 0x401eafa0 4>
                                align 32 context <function_decl 0x402fc7e0 atoll>
                                abstract_origin <parm_decl 0x40310310 __base>
                                initial <integer_cst 0x4030f5c0 10>>
                            chain <decl_stmt 0x4037fe88
                                arg 0 <var_decl 0x4039ba10
                                    type <integer_type 0x401da540 long long int>
                                    used DI file /usr/include/stdlib.h line 343
                                    size <integer_cst 0x401eafc0 64>
                                    unit size <integer_cst 0x401ed0e0 8>
                                    align 64
                                    context <function_decl 0x402fc7e0 atoll>
                                    abstract_origin <result_decl 0x40310460>>
                                chain <compound_stmt 0x4037fe9c tree_2
                                    arg 0 <scope_stmt 0x4037feb0 tree_0
                                        chain <expr_stmt 0x4037fed8 tree_1
                                            arg 0 <modify_expr 0x4039fc60>
                                            chain <goto_stmt 0x4037ff50 tree_0>>>
                                    chain <scope_stmt 0x4037ff78 tree_3
                                        arg 0 <block 0x40397e40>
                                        chain <label_stmt 0x4037ff8c
                                            arg 0 <label_decl 0x4039b9a0>
                                            chain <expr_stmt 0x4037fe74 addressable
                                                arg 0 <var_decl 0x4039ba10>>>>>>>>>>>
        arg 1 <identifier_node 0x403979c0 /usr/include/stdlib.h>
        /usr/include/stdlib.h:343:0>
```

SWITCH_EXPR e 2

Switch 표현식.
Operand 0 는 branch 를 실행하는데 사용되는 표현식이다.
Operand 1 는 case 값들을 포함한다. 우리가 조직화한 방식은 정의된 front-end implementation 이다.

C 에서 사용되지 않는 TREE node 입니다.

EXC_PTR_EXPR e 0

실시간으로부터 예외 object.

C 에서 사용되지 않는 TREE node 입니다.

제 5 절 tree 에 사용되는 전역변수들과 열거자

5.1 열거자

enum tree_code

tree node 들을 위한 code 들을 열거자 형태로 가지고 있다.

나의 환경에서는 아래와 같이 설정이 된다. 아래를 보면 위에서 설명한 하나의 요소요소를 나열한 것을 알수 있다.

```
enum tree_code
{
  ERROR_MARK, IDENTIFIER_NODE, TREE_LIST, TREE_VEC, BLOCK, VOID_TYPE,
  INTEGER_TYPE, REAL_TYPE, COMPLEX_TYPE, VECTOR_TYPE, ENUMERAL_TYPE,
  BOOLEAN_TYPE, CHAR_TYPE, POINTER_TYPE, OFFSET_TYPE, REFERENCE_TYPE,
  METHOD_TYPE, FILE_TYPE, ARRAY_TYPE, SET_TYPE, RECORD_TYPE, UNION_TYPE,
  QUAL_UNION_TYPE, FUNCTION_TYPE, LANG_TYPE, INTEGER_CST, REAL_CST, COMPLEX_CST,
  VECTOR_CST, STRING_CST, FUNCTION_DECL, LABEL_DECL, CONST_DECL, TYPE_DECL,
  VAR_DECL, PARM_DECL, RESULT_DECL, FIELD_DECL, NAMESPACE_DECL, COMPONENT_REF,
  BIT_FIELD_REF, INDIRECT_REF, BUFFER_REF, ARRAY_REF, ARRAY_RANGE_REF,
  VTABLE_REF, CONSTRUCTOR, COMPOUND_EXPR, MODIFY_EXPR, INIT_EXPR, TARGET_EXPR,
  COND_EXPR, BIND_EXPR, CALL_EXPR, METHOD_CALL_EXPR, WITH_CLEANUP_EXPR,
  CLEANUP_POINT_EXPR, PLACEHOLDER_EXPR, WITH_RECORD_EXPR, PLUS_EXPR, MINUS_EXPR,
  MULT_EXPR, TRUNC_DIV_EXPR, CEIL_DIV_EXPR, FLOOR_DIV_EXPR, ROUND_DIV_EXPR,
  TRUNC_MOD_EXPR, CEIL_MOD_EXPR, FLOOR_MOD_EXPR, ROUND_MOD_EXPR, RDIV_EXPR,
  EXACT_DIV_EXPR, FIX_TRUNC_EXPR, FIX_CEIL_EXPR, FIX_FLOOR_EXPR,
  FIX_ROUND_EXPR, FLOAT_EXPR, NEGATE_EXPR, MIN_EXPR, MAX_EXPR, ABS_EXPR,
  FFS_EXPR, LSHIFT_EXPR, RSHIFT_EXPR, LROTATE_EXPR, RROTATE_EXPR, BIT_IOR_EXPR,
  BIT_XOR_EXPR, BIT_AND_EXPR, BIT_ANDTC_EXPR, BIT_NOT_EXPR, TRUTH_ANDIF_EXPR,
  TRUTH_ORIF_EXPR,TRUTH_AND_EXPR, TRUTH_OR_EXPR, TRUTH_XOR_EXPR,
  TRUTH_NOT_EXPR, LT_EXPR, LE_EXPR, GT_EXPR, GE_EXPR , EQ_EXPR, NE_EXPR,
  UNORDERED_EXPR, ORDERED_EXPR, UNLT_EXPR, UNLE_EXPR, UNGT_EXPR, UNGE_EXPR,
  UNEQ_EXPR, IN_EXPR, SET_LE_EXPR, CARD_EXPR, RANGE_EXPR, CONVERT_EXPR,
  NOP_EXPR, NON_LVALUE_EXPR, VIEW_CONVERT_EXPR, SAVE_EXPR, UNSAVE_EXPR,
  RTL_EXPR, ADDR_EXPR, REFERENCE_EXPR, ENTRY_VALUE_EXPR, FDESC_EXPR,
  COMPLEX_EXPR, CONJ_EXPR, REALPART_EXPR, IMAGPART_EXPR, PREDECREMENT_EXPR,
  PREINCREMENT_EXPR , POSTDECREMENT_EXPR, POSTINCREMENT_EXPR,VA_ARG_EXPR,
  TRY_CATCH_EXPR, TRY_FINALLY_EXPR, GOTO_SUBROUTINE_EXPR, LABEL_EXPR,
  GOTO_EXPR, RETURN_EXPR, EXIT_EXPR, LOOP_EXPR, LABELED_BLOCK_EXPR,
  EXIT_BLOCK_EXPR, EXPR_WITH_FILE_LOCATION, SWITCH_EXPR, EXC_PTR_EXPR,
  LAST_AND_UNUSED_TREE_CODE
};
```

enum built_in_class

컴파일러의 어떤 부분이 주어진 builtin 함수를 정의하고 있는지 분류한다. 참고 : 우리는 이것이 2 비트 이상이 아닐거라고 가정하고 있다.

```
enum built_in_class
{
  NOT_BUILT_IN = 0,
  BUILT_IN_FRONTEND,
  BUILT_IN_MD,
  BUILT_IN_NORMAL
};
```

enum built_in_function

함수들내의 여러 built 를 구별해내기 위한 code 들, 그래서 expand_call 는 그들을 빨리 구별할 수 있다.

```
enum built_in_function
{
  BUILT_IN_ALLOCA, BUILT_IN_ABS, BUILT_IN_LABS, BUILT_IN_FABS, BUILT_IN_FABSF,
  BUILT_IN_FABSL, BUILT_IN_LLABS, BUILT_IN_IMAXABS, BUILT_IN_CONJ,
  BUILT_IN_CONJF, BUILT_IN_CONJL, BUILT_IN_CREAL, BUILT_IN_CREALF,
  UILT_IN_CREALL, BUILT_IN_CIMAG, BUILT_IN_CIMAGF, BUILT_IN_CIMAGL, BUILT_IN_DIV,
  BUILT_IN_LDIV, BUILT_IN_FFLOOR, BUILT_IN_FCEIL, BUILT_IN_FMOD, BUILT_IN_FREM,
  BUILT_IN_BZERO, BUILT_IN_BCMP, BUILT_IN_FFS, BUILT_IN_INDEX, BUILT_IN_RINDEX,
  BUILT_IN_MEMCPY, BUILT_IN_MEMCMP,
  BUILT_IN_MEMSET, BUILT_IN_STRCAT, BUILT_IN_STRNCAT, BUILT_IN_STRCPY,
  BUILT_IN_STRNCPY, BUILT_IN_STRCMP, BUILT_IN_STRNCMP, BUILT_IN_STRLEN,
  BUILT_IN_STRSTR, BUILT_IN_STRPBRK, BUILT_IN_STRSPN, BUILT_IN_STRCSPN,
  BUILT_IN_STRCHR, BUILT_IN_STRRCHR, BUILT_IN_SQRT, BUILT_IN_SIN, BUILT_IN_COS,
  BUILT_IN_SQRTF, BUILT_IN_SINF, BUILT_IN_COSF, BUILT_IN_SQRTL, BUILT_IN_SINL,
  BUILT_IN_COSL, BUILT_IN_GETEXP, BUILT_IN_GETMAN, BUILT_IN_SAVEREGS,
  BUILT_IN_CLASSIFY_TYPE, BUILT_IN_NEXT_ARG, BUILT_IN_ARGS_INFO,
  BUILT_IN_CONSTANT_P, BUILT_IN_FRAME_ADDRESS, BUILT_IN_RETURN_ADDRESS,
  BUILT_IN_AGGREGATE_INCOMING_ADDRESS, BUILT_IN_APPLY_ARGS, BUILT_IN_APPLY,
  BUILT_IN_RETURN, BUILT_IN_SETJMP, BUILT_IN_LONGJMP, BUILT_IN_TRAP,
  BUILT_IN_PREFETCH, BUILT_IN_PUTCHAR, BUILT_IN_PUTS, BUILT_IN_PRINTF,
  BUILT_IN_FPUTC, BUILT_IN_FPUTS, BUILT_IN_FWRITE, BUILT_IN_FPRINTF,
  BUILT_IN_PUTCHAR_UNLOCKED, BUILT_IN_PUTS_UNLOCKED, BUILT_IN_PRINTF_UNLOCKED,
  BUILT_IN_FPUTC_UNLOCKED, BUILT_IN_FPUTS_UNLOCKED, BUILT_IN_FWRITE_UNLOCKED,
  BUILT_IN_FPRINTF_UNLOCKED, BUILT_IN_ISGREATER, BUILT_IN_ISGREATEREQUAL,
  BUILT_IN_ISLESS, BUILT_IN_ISLESSEQUAL, BUILT_IN_ISLESSGREATER,
  BUILT_IN_ISUNORDERED, BUILT_IN_UNWIND_INIT, BUILT_IN_DWARF_CFA,
  BUILT_IN_DWARF_FP_REGNUM, BUILT_IN_INIT_DWARF_REG_SIZES,
  BUILT_IN_FROB_RETURN_ADDR, BUILT_IN_EXTRACT_RETURN_ADDR, BUILT_IN_EH_RETURN,
  BUILT_IN_EH_RETURN_DATA_REGNO, BUILT_IN_VARARGS_START, BUILT_IN_STDARG_START,
  BUILT_IN_VA_END, BUILT_IN_VA_COPY, BUILT_IN_EXPECT, BUILT_IN_NEW,
  BUILT_IN_VEC_NEW, BUILT_IN_DELETE, BUILT_IN_VEC_DELETE, END_BUILTINS
};
```

5.2 전역변수와 접근 매크로

tree global_trees[TI_MAX]

C 컴파일러의 명명된(named) 혹은 명명되지 않은 표준 data type 을 미리 가지고 있는 변수이다.

아래는 이 변수가 어떻게 선언되어 있고 접근을 위해서 사용되는 매크로에 대한 것이다.

```
enum tree_index {
  TI_ERROR_MARK,
  TI_INTQI_TYPE,
  TI_INTHI_TYPE,
  TI_INTSI_TYPE,
  TI_INTDI_TYPE,
  TI_INTTI_TYPE,

  TI_UINTQI_TYPE,
  TI_UINTHI_TYPE,
  TI_UINTSI_TYPE,
  TI_UINTDI_TYPE,
  TI_UINTTI_TYPE,

  TI_INTEGER_ZERO,
  TI_INTEGER_ONE,
  TI_INTEGER_MINUS_ONE,
  TI_NULL_POINTER,

  TI_SIZE_ZERO,
  TI_SIZE_ONE,

  TI_BITSIZE_ZERO,
  TI_BITSIZE_ONE,
  TI_BITSIZE_UNIT,

  TI_COMPLEX_INTEGER_TYPE,
  TI_COMPLEX_FLOAT_TYPE,
  TI_COMPLEX_DOUBLE_TYPE,
  TI_COMPLEX_LONG_DOUBLE_TYPE,

  TI_FLOAT_TYPE,
  TI_DOUBLE_TYPE,
  TI_LONG_DOUBLE_TYPE,

  TI_VOID_TYPE,
  TI_PTR_TYPE,
  TI_CONST_PTR_TYPE,
  TI_PTRDIFF_TYPE,
  TI_VA_LIST_TYPE,

  TI_VOID_LIST_NODE,

  TI_UV4SF_TYPE,
  TI_UV4SI_TYPE,
  TI_UV8HI_TYPE,
  TI_UV8QI_TYPE,
  TI_UV4HI_TYPE,
  TI_UV2SI_TYPE,
  TI_UV2SF_TYPE,
  TI_UV16QI_TYPE,
```

```
  TI_V4SF_TYPE,
  TI_V16SF_TYPE,
  TI_V4SI_TYPE,
  TI_V8HI_TYPE,
  TI_V8QI_TYPE,
  TI_V4HI_TYPE,
  TI_V2SI_TYPE,
  TI_V2SF_TYPE,
  TI_V16QI_TYPE,

  TI_MAIN_IDENTIFIER,

  TI_MAX
};
```

이것은 각 배열이 가지는 의미를 위한 열거자이다.

```
#define error_mark_node         global_trees[TI_ERROR_MARK]

#define intQI_type_node         global_trees[TI_INTQI_TYPE]
#define intHI_type_node         global_trees[TI_INTHI_TYPE]
#define intSI_type_node         global_trees[TI_INTSI_TYPE]
#define intDI_type_node         global_trees[TI_INTDI_TYPE]
#define intTI_type_node         global_trees[TI_INTTI_TYPE]

#define unsigned_intQI_type_node    global_trees[TI_UINTQI_TYPE]
#define unsigned_intHI_type_node    global_trees[TI_UINTHI_TYPE]
#define unsigned_intSI_type_node    global_trees[TI_UINTSI_TYPE]
#define unsigned_intDI_type_node    global_trees[TI_UINTDI_TYPE]
#define unsigned_intTI_type_node    global_trees[TI_UINTTI_TYPE]

#define integer_zero_node       global_trees[TI_INTEGER_ZERO]
#define integer_one_node        global_trees[TI_INTEGER_ONE]
#define integer_minus_one_node  \
                                global_trees[TI_INTEGER_MINUS_ONE]
#define size_zero_node          global_trees[TI_SIZE_ZERO]
#define size_one_node           global_trees[TI_SIZE_ONE]
#define bitsize_zero_node       global_trees[TI_BITSIZE_ZERO]
#define bitsize_one_node        global_trees[TI_BITSIZE_ONE]
#define bitsize_unit_node       global_trees[TI_BITSIZE_UNIT]

#define null_pointer_node       global_trees[TI_NULL_POINTER]

#define float_type_node         global_trees[TI_FLOAT_TYPE]
#define double_type_node        global_trees[TI_DOUBLE_TYPE]
#define long_double_type_node   \
                                global_trees[TI_LONG_DOUBLE_TYPE]

#define complex_integer_type_node   \
                    global_trees[TI_COMPLEX_INTEGER_TYPE]
#define complex_float_type_node     \
                    global_trees[TI_COMPLEX_FLOAT_TYPE]
```

```
#define complex_double_type_node    \
                    global_trees[TI_COMPLEX_DOUBLE_TYPE]
#define complex_long_double_type_node   \
                    global_trees[TI_COMPLEX_LONG_DOUBLE_TYPE]

#define void_type_node          global_trees[TI_VOID_TYPE]
#define ptr_type_node           global_trees[TI_PTR_TYPE]
#define const_ptr_type_node     global_trees[TI_CONST_PTR_TYPE]
#define ptrdiff_type_node       global_trees[TI_PTRDIFF_TYPE]
#define va_list_type_node       global_trees[TI_VA_LIST_TYPE]

#define void_list_node          global_trees[TI_VOID_LIST_NODE]

#define main_identifier_node        \
                    global_trees[TI_MAIN_IDENTIFIER]

#define unsigned_V16QI_type_node    global_trees[TI_UV16QI_TYPE]
#define unsigned_V4SI_type_node     global_trees[TI_UV4SI_TYPE]
#define unsigned_V8QI_type_node     global_trees[TI_UV8QI_TYPE]
#define unsigned_V8HI_type_node     global_trees[TI_UV8HI_TYPE]
#define unsigned_V4HI_type_node     global_trees[TI_UV4HI_TYPE]
#define unsigned_V2SI_type_node     global_trees[TI_UV2SI_TYPE]

#define V16QI_type_node         global_trees[TI_V16QI_TYPE]
#define V4SF_type_node          global_trees[TI_V4SF_TYPE]
#define V4SI_type_node          global_trees[TI_V4SI_TYPE]
#define V8QI_type_node          global_trees[TI_V8QI_TYPE]
#define V8HI_type_node          global_trees[TI_V8HI_TYPE]
#define V4HI_type_node          global_trees[TI_V4HI_TYPE]
#define V2SI_type_node          global_trees[TI_V2SI_TYPE]
#define V2SF_type_node          global_trees[TI_V2SF_TYPE]
#define V16SF_type_node         global_trees[TI_V16SF_TYPE]
```

위에서의 intQI_type_node, intHI_type_node, intSI_type_node, intDI_type_node, intTI_type_node 와 unsigned_intQI_type_node, unsigned_intHI_type_node, unsigned_intSI_type_node, unsigned_intDI_type_node, unsigned_intTI_type_node 는 $prefix/gcc/tree.c 의 build_common_tree_nodes () 함수내에서 설정된다.

tree integer_types[itk_none]

```
/* 표준 C 정수 type 에 대한 열거. 이것은 짧은 type 에서 긴 것 순으로
   나열되어야 합니다.  */
enum integer_type_kind
{
  itk_char,
  itk_signed_char,
  itk_unsigned_char,
  itk_short,
  itk_unsigned_short,
  itk_int,
  itk_unsigned_int,
  itk_long,
  itk_unsigned_long,
```

```
  itk_long_long,
  itk_unsigned_long_long,
  itk_none
};

extern tree integer_types[itk_none];

#define char_type_node                  integer_types[itk_char]
#define signed_char_type_node           integer_types[itk_signed_char]
#define unsigned_char_type_node         integer_types[itk_unsigned_char]
#define short_integer_type_node         integer_types[itk_short]
#define short_unsigned_type_node        integer_types[itk_unsigned_short]
#define integer_type_node               integer_types[itk_int]
#define unsigned_type_node              integer_types[itk_unsigned_int]
#define long_integer_type_node          integer_types[itk_long]
#define long_unsigned_type_node         integer_types[itk_unsigned_long]
#define long_long_integer_type_node     integer_types[itk_long_long]
#define long_long_unsigned_type_node    integer_types[itk_unsigned_long_long]
```

위의 모든 변수들은 $prefix/gcc/tree.c 의 build_common_tree_nodes () 함수내에서 설정된다.

char tree_code_type[MAX_TREE_CODES]

tree code 들을 분류하는 문자를 내용물로 가지고 있다. 아래 한 문자에 대한 의미는 이미 4.1 장에서 소개하였다.

```
#define MAX_TREE_CODES 256
char tree_code_type[MAX_TREE_CODES] =
{
  x x x x b t t t t t t t t t t t
  t t t t t t t t t c c c c c d d
  d d d d d d d r r r r r r r e e
  e e e e e e e e e x e 2 2 2 2 2
  2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2
  1 1 2 2 2 2 2 2 2 2 1 e e e e e
  e < < < < < < < < < < < < < 2 <
  1 2 1 1 1 1 e e e e e e e 2 1 1
  1 e e e e e e e e s s s s s e e
  e e e
};
```

char tree_code_length[MAX_TREE_CODES]

tree code 순서대로 나열된 expression operand 의 수에 관한 테이블. 이 operand 는 node structure 의 고정된 부분이다. 하지만 type 들과 decl 들에는 사용되지 않는다.

아래와 같이 설정된다.

```
int tree_code_length[MAX_TREE_CODES] =
{
  0 11 2 2 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 2 3 3 3 3 0 0
  0 0 0 0 0 0 0 2 3 1 1 2 2 3 2 2
  2 2 4 3 3 2 4 3 1 0 2 2 2 2 2 2
  2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2
```

```
  1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2
  1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  1 2 1 1 1 1 3 1 2 1 1 1 2 2 1 1
  1 2 2 2 2 1 2 2 2 1 1 1 1 1 2 2
  3 2 0
};
```

여기서 살펴봐야 할 것이 있다면 두번째 "identifier_node" 의 값은 일반적으로는 -1 값을 가지고 있는데, 11 로 변화가 되었다. 이 값은 $prefix/gcc/toplev.c 파일내 lang_independent_init () 함수에서 업데이트된다.

```
tree_code_length[(int) IDENTIFIER_NODE]
  = ((lang_hooks.identifier_size - sizeof (struct tree_common)
      + sizeof (tree) - 1) / sizeof (tree));
```

char tree_code_name[MAX_TREE_CODES]

tree component 들의 이름들. tree 를 출력하거나 오류 메세지를 출력할 때 사용된다.

```
int tree_code_name[MAX_TREE_CODES] =
{
  "error_mark",
  "identifier_node", "tree_list", "tree_vec", "block", "void_type",
  "integer_type", "real_type", "complex_type", "vector_type",
  "enumeral_type", "boolean_type", "char_type", "pointer_type",
  "offset_type", "reference_type", "method_type", "file_type",
  "array_type", "set_type", "record_type", "union_type",
  "qual_union_type", "function_type", "lang_type", "integer_cst",
  "real_cst", "complex_cst", "vector_cst", "string_cst",
  "function_decl", "label_decl", "const_decl", "type_decl",
  "var_decl", "parm_decl", "result_decl", "field_decl",
  "namespace_decl", "component_ref", "bit_field_ref",
  "indirect_ref", "buffer_ref", "array_ref", "array_range_ref",
  "vtable_ref", "constructor", "compound_expr", "modify_expr",
  "init_expr", "target_expr", "cond_expr", "bind_expr", "call_expr",
  "method_call_expr", "with_cleanup_expr", "cleanup_point_expr",
  "placeholder_expr", "with_record_expr", "plus_expr", "minus_expr",
  "mult_expr", "trunc_div_expr", "ceil_div_expr", "floor_div_expr",
  "round_div_expr", "trunc_mod_expr", "ceil_mod_expr",
  "floor_mod_expr", "round_mod_expr", "rdiv_expr", "exact_div_expr",
  "fix_trunc_expr", "fix_ceil_expr", "fix_floor_expr",
  "fix_round_expr", "float_expr", "negate_expr", "min_expr",
  "max_expr", "abs_expr", "ffs_expr", "lshift_expr", "rshift_expr",
  "lrotate_expr", "rrotate_expr", "bit_ior_expr", "bit_xor_expr",
  "bit_and_expr", "bit_andtc_expr", "bit_not_expr",
  "truth_andif_expr", "truth_orif_expr", "truth_and_expr",
  "truth_or_expr", "truth_xor_expr", "truth_not_expr", "lt_expr",
  "le_expr", "gt_expr", "ge_expr", "eq_expr", "ne_expr",
  "unordered_expr", "ordered_expr", "unlt_expr", "unle_expr",
  "ungt_expr", "unge_expr", "uneq_expr", "in_expr", "set_le_expr",
  "card_expr", "range_expr", "convert_expr", "nop_expr",
  "non_lvalue_expr", "view_convert_expr", "save_expr",
  "unsave_expr", "rtl_expr", "addr_expr", "reference_expr",
```

```
  "entry_value_expr", "fdesc_expr", "complex_expr", "conj_expr",
  "realpart_expr", "imagpart_expr", "predecrement_expr",
  "preincrement_expr", "postdecrement_expr", "postincrement_expr",
  "va_arg_expr", "try_catch_expr", "try_finally", "goto_subroutine",
  "label_expr", "goto_expr", "return_expr", "exit_expr",
  "loop_expr", "labeled_block_expr", "exit_block_expr",
  "expr_with_file_location", "switch_expr", "exc_ptr_expr"
};
```

static int next_decl_uid

다음 tree decl 이 생성될 때 사용될 유일한 ID

static int next_type_uid

다음 tree type 이 생성될 때 사용될 유일한 ID

tree sizetype_tab[(int) TYPE_KIND_LAST]

```
/* Data 구조체들과 매크로들, size 들을 다룰 함수들, size 를 나타내는데
   사용되는 여러 type 들을 정의한다. */

enum size_type_kind
{
  SIZETYPE,             /* 바이트인 size 들의 일반적인 표시. */
  SSIZETYPE,            /* 바이트인 size 들의 Sign 인 표시. */
  USIZETYPE,            /* 바이트인 size 들의 Unsign 인 표시. */
  BITSIZETYPE,          /* 비트인 size 들의 일반적인 표시. */
  SBITSIZETYPE,         /* 비트인 size 들의 Sign 인 표시. */
  UBITSIZETYPE,         /* 비트인 size 들의 Unsign 인 표시. */
  TYPE_KIND_LAST};

#define sizetype sizetype_tab[(int) SIZETYPE]
#define bitsizetype sizetype_tab[(int) BITSIZETYPE]
#define ssizetype sizetype_tab[(int) SSIZETYPE]
#define usizetype sizetype_tab[(int) USIZETYPE]
#define sbitsizetype sizetype_tab[(int) SBITSIZETYPE]
#define ubitsizetype sizetype_tab[(int) UBITSIZETYPE]
```

여기서 sizetype 과 bitsizetype 은 initialize_sizetypes () 함수에서 초기화 된다.

tree c_global_trees[CTI_MAX]

```
/* C 컴파일러의 명명된 혹은 명명되지 않는 표준 data type 들. */

enum c_tree_index
{
    CTI_WCHAR_TYPE,
    CTI_SIGNED_WCHAR_TYPE,
    CTI_UNSIGNED_WCHAR_TYPE,
    CTI_WINT_TYPE,
    CTI_C_SIZE_TYPE, /* size_t typedef 와 sizeof 의 결과 type
                        (내부 sizetype 과는 달리 TYPE_IS_SIZETYPE
                        설정이 없는 보통의 type) 에 사용되는 type. */
    CTI_SIGNED_SIZE_TYPE, /* Format checking 만을 위해. */
```

```
    CTI_UNSIGNED_PTRDIFF_TYPE, /* Format checking 만을 위해.  */
    CTI_INTMAX_TYPE,
    CTI_UINTMAX_TYPE,
    CTI_WIDEST_INT_LIT_TYPE,
    CTI_WIDEST_UINT_LIT_TYPE,

    CTI_CHAR_ARRAY_TYPE,
    CTI_WCHAR_ARRAY_TYPE,
    CTI_INT_ARRAY_TYPE,
    CTI_STRING_TYPE,
    CTI_CONST_STRING_TYPE,

    /* Boolean 표현식들을 위한 type (C++ 에서는 bool, C 에서는 int).  */
    CTI_BOOLEAN_TYPE,
    CTI_BOOLEAN_TRUE,
    CTI_BOOLEAN_FALSE,
    /* C99 의 _Bool type.  */
    CTI_C_BOOL_TYPE,
    CTI_C_BOOL_TRUE,
    CTI_C_BOOL_FALSE,
    CTI_DEFAULT_FUNCTION_TYPE,

    CTI_G77_INTEGER_TYPE,
    CTI_G77_UINTEGER_TYPE,
    CTI_G77_LONGINT_TYPE,
    CTI_G77_ULONGINT_TYPE,

    /* type 들은 없지만 우리는 이것들을 항상 반드시 지켜봐야 한다.  */
    CTI_FUNCTION_NAME_DECL,
    CTI_PRETTY_FUNCTION_NAME_DECL,
    CTI_C99_FUNCTION_NAME_DECL,
    CTI_SAVED_FUNCTION_NAME_DECLS,

    CTI_VOID_ZERO,

    CTI_MAX
};

#define wchar_type_node                 c_global_trees[CTI_WCHAR_TYPE]
#define signed_wchar_type_node          c_global_trees[CTI_SIGNED_WCHAR_TYPE]
#define unsigned_wchar_type_node        c_global_trees[CTI_UNSIGNED_WCHAR_TYPE]
#define wint_type_node                  c_global_trees[CTI_WINT_TYPE]
#define c_size_type_node                c_global_trees[CTI_C_SIZE_TYPE]
#define signed_size_type_node           c_global_trees[CTI_SIGNED_SIZE_TYPE]
#define unsigned_ptrdiff_type_node      c_global_trees[CTI_UNSIGNED_PTRDIFF_TYPE]
#define intmax_type_node                c_global_trees[CTI_INTMAX_TYPE]
#define uintmax_type_node               c_global_trees[CTI_UINTMAX_TYPE]
#define widest_integer_literal_type_node        \
                                        c_global_trees[CTI_WIDEST_INT_LIT_TYPE]
#define widest_unsigned_literal_type_node       \
                                        c_global_trees[CTI_WIDEST_UINT_LIT_TYPE]
```

```
#define boolean_type_node               c_global_trees[CTI_BOOLEAN_TYPE]
#define boolean_true_node               c_global_trees[CTI_BOOLEAN_TRUE]
#define boolean_false_node              c_global_trees[CTI_BOOLEAN_FALSE]

#define c_bool_type_node                c_global_trees[CTI_C_BOOL_TYPE]
#define c_bool_true_node                c_global_trees[CTI_C_BOOL_TRUE]
#define c_bool_false_node               c_global_trees[CTI_C_BOOL_FALSE]

#define char_array_type_node            c_global_trees[CTI_CHAR_ARRAY_TYPE]
#define wchar_array_type_node           c_global_trees[CTI_WCHAR_ARRAY_TYPE]
#define int_array_type_node             c_global_trees[CTI_INT_ARRAY_TYPE]
#define string_type_node                c_global_trees[CTI_STRING_TYPE]
#define const_string_type_node          c_global_trees[CTI_CONST_STRING_TYPE]

#define default_function_type           c_global_trees[CTI_DEFAULT_FUNCTION_TYPE]

/* 반드시 반드시 f/com.h 와 sync 가 유지되어야 하는 g77 integer type 들 */
#define g77_integer_type_node           c_global_trees[CTI_G77_INTEGER_TYPE]
#define g77_uinteger_type_node          c_global_trees[CTI_G77_UINTEGER_TYPE]
#define g77_longint_type_node           c_global_trees[CTI_G77_LONGINT_TYPE]
#define g77_ulongint_type_node          c_global_trees[CTI_G77_ULONGINT_TYPE]

#define function_name_decl_node         c_global_trees[CTI_FUNCTION_NAME_DECL]
#define pretty_function_name_decl_node          \
                                c_global_trees[CTI_PRETTY_FUNCTION_NAME_DECL]
#define c99_function_name_decl_node             \
                                c_global_trees[CTI_C99_FUNCTION_NAME_DECL]
#define saved_function_name_decls               \
                                c_global_trees[CTI_SAVED_FUNCTION_NAME_DECLS]

/* '((void) 0)' 를 위한 node.  */
#define void_zero_node                  c_global_trees[CTI_VOID_ZERO]
```

const char *const built_in_names[(int) END_BUILTINS];

enum built_in_function 의 실제 이름 관련 문자열을 가지고 있는 전역 변수.

tree built_in_decls[(int) END_BUILTINS];

enum built_in_function 를 위한 _DECL_ tree 들의 배열.

제 6 절 tree 를 위한 부과적인 구조체

통계 수집을 위한 변수들.

```
typedef enum
{
  d_kind,
  t_kind,
  b_kind,
  s_kind,
  r_kind,
  e_kind,
```

```
  c_kind,
  id_kind,
  perm_list_kind,
  temp_list_kind,
  vec_kind,
  x_kind,
  lang_decl,
  lang_type,
  all_kinds
} tree_node_kind;

int tree_node_counts[(int) all_kinds];
int tree_node_sizes[(int) all_kinds];

static const char * const tree_node_kind_names[] = {
  "decls",
  "types",
  "blocks",
  "stmts",
  "refs",
  "exprs",
  "constants",
  "identifiers",
  "perm_tree_lists",
  "temp_tree_lists",
  "vecs",
  "random kinds",
  "lang_decl kinds",
  "lang_type kinds"
};
```

제 7 절 tree 를 위한 함수들

이 절에서는 $prefix/gcc/tree.c 파일에 선언되어 있는 함수들을 모두 살펴 보도록 하자. 이 함수의 배열 순서는 알파벳순이 아니라, 함수가 선언된 순어에 따라 배열되어 있음을 알기 바랍니다. 후에 이에 대한 정렬 요청이 있을 경우, 알파벳 순을 만들도록 하겠습니다.

void set_decl_assembler_name (tree *decl*)

DECL 을 위한 DECL_ASSEMBLER_NAME 를 설정.

void init_obstacks ()

principal obstack 를 초기화합니다.

char * permalloc (int *size*)

Permanent obstack 에 SIZE 바이트를 할당하고 그것에 대한 포인터를 반환한다.

char * perm_calloc (int *nelem*, long *size*)

Permanent obstack 에 SIZE 바이트 크기인 NELEM 개의 아이템을 할당하고 그에 대한 포인터를 반환한다. 저장소는 값을 반환하기 전에 깨끗히 된다.

size_t tree_size (tree node)

> 'node' 에 의해 채용될 바이트의 수를 계산합니다. 이 루틴은 단지 TREE_CODE 만 살펴보며, 만약 code 가 TREE_VEC 이면 TREE_VEC_LENGTH 를 살펴본다.

tree make_node (enum tree_code *code*)

> code 가 CODE 인 새로이 할당된 node 를 되돌립니다. decl 와 type node 들을 위해 몇몇 다른 field 들은 초기화합니다. node 의 나머지는 0 으로 초기화 합니다.

tree make_lang_type (enum tree_code code)

> 'node' 에 의해 채용될 바이트의 수를 계산합니다. 이 루틴은 단지 TREE_CODE 만 살펴보며, 만약 code 가 TREE_VEC 이면 TREE_VEC_LENGTH 를 살펴본다.

tree copy_node (tree node)

> NODE 의 TREE_CHAIN 이 0 이고 이것은 새로운 uid 를 가진다는 것을 제외하고는 같은 내용을 가진 새로운 node 를 반환한다.

tree copy_list (tree list)

> TREE_CHAIN field 를 통해 연결된 node 들의 chain 의 복사물을 반환한다. 예를 들면, 이것은 TREE_LIST node 들로 만들어진 한 list 를 복사할 수 있다.

tree build_int_2_wide (unsigned HOST_WIDE_INT low, HOST_WIDE_INT hi)

> 상수 값이 두개의 정수 LOW 와 HI 로 지정된 새로이 만들어진 INTERGET_CST node 를 반환한다.
> TREE_TYPE 은 'int' 로 설정되었다.
>
> 이 함수는 'build_int_2' 매크로를 통해서 반드시 호출되어야 한다.

tree build_vector (tree type, tree vals)

> Type 이 TYPE 이고 값이 VALS 에 지정된 list 로 된 새로운 VECTOR_CST node 를 반환한다.

tree build_real (tree type, REAL_VALUE_TYPE d)

> Type 이 TYPE 이고 값이 D 인 새로운 REAL_CST node 를 반환한다.

REAL_VALUE_TYPE real_value_from_int_cst (tree type ATTRIBUTE_UNUSED, tree i);

> Type 이 TYPE 이고 값이 INTERGER_CST node I 의 정수값인 새로운 REAL_CST node 를 반환한다.

```
static void
build_real_from_int_cst_1 (PTR data)
```

> 비록 floating point exception handler 에 의해 보호되고 있지만 integer 를 floting point 로 변환한다.

```
tree
build_real_from_int_cst (tree type, tree i)
```

> 정수 상수 I 를 나타내는 주어진 tree 를 사용하여, Type 이 TYPE 인 같은 값을 가지는 floating-point 상수를 나타내는 tree 를 반환한다. 만약 floating-point 값들상의 산술 연산을 행할 방법이 없을 경우 이 수행을 행할 수 없다.

```
tree
build_string (int len, const chat *str)
```

값이 LEN 길이의 문자열 STR 을 가지는 새로이 만들어진 STRING_CST node 를 반환한다. TREE_TYPE 은 초기화되지 않는다.

```
tree
build_complex (tree type, tree real, tree imag)
```

실수와 허수 부분인 REAL 과 IMAG 에 의해 지정된 값을 가지는 새로이 만들어진 COMPLEX_CST node 를 반환한다. REAL 과 IMAG 둘다 반드시 상수 node 들이여야 한다. 만약 TYPE 이 지정될 경우, 이것은 COMPLEX_CST 의 type 이여야하며; 지정이 안될 경우 새로운 type 이 만들어질 것이다.

```
tree
make_tree_vec (int len)
```

길이 LEN 의 새로이 만들어진 TREE_VEC node 를 생성한다.

```
int
integer_zerop (tree expr)
```

만약 EXPR 이 정수 상수 0 혹은 복소수 상수 0 일 경우 1 을 반환한다.

```
int
integer_onep (tree expr)
```

만약 EXPR 이 정수 상수 1 혹은 대응하는 복소수 상수일 경우 1 을 반환한다.

```
int
integer_all_onesp (tree expr)
```

만약 EXPR 가 포함하는 가장 큰 precision 내에 모두 1 들로 채워져 있는 정 수라면 1 을 반환한다. 마찬가지로 복소수 상수에 대해서도 비슷하다.

```
int
integer_pow2p (tree expr)
```

만약 EXPR 이 2 의 제곱인 정수 상수일 경우 (예를 들면, 하나의 비트만 설정된) 1 을 반환한다.

```
int
tree_log2 (tree expr)
```

2 의 배수로 알려져 있는 tree node 에 의해 나타나는 2 의 배수를 반환한다.

```
int
tree_floor_log2 (tree expr)
```

비슷하지만, 2 ** Y 가 EXPR 과 같거나 적은, 가장 큰 정수 Y 를 반환한다.

```
int
real_zerop (tree expr)
```

만약 EXPR 이 실수 상수 0 이면 1 을 반환한다.

```
int
real_onep (tree expr)
```

만약 EXPR 이 실수 상수 1 이거나 실수 혹은 복소수 형태일 경우 1 을 반환한다.

```
int
real_twop (tree expr)
```

만약 EXPR 이 실수 상수 2 일 경우 1 을 반환한다.

```
int
really_constant_p (tree exp)
```

만약 EXP 가 상수 혹은 상수의 cast 이면 0 이 아닌 값 반환.

```
tree
value_member (tree elem, tree list)
```

TREE_VALUE 가 ELEM 인 처음 list element 를 반환하다. 만약 ELEM 이 LIST 상에 없다면 0 을 반환한다.

```
tree
purpose_member (tree elem, tree list)
```

TREE_PURPOSE 가 ELEM 인 처음 list element 를 반환한다. 만약 ELEM 이 LIST 상에 없다면 0 을 반환한다.

```
tree
binfo_member (tree elem, tree list)
```

BINFO_TYPE 이 ELEM 인 처음 list element 를 반환한다. 만약 ELEM 이 LIST 상에 없다면 0 을 반환한다.

```
int
chain_member (tree elem, tree chain)
```

만약 ELEM 이 chain CHAIN 의 부분이면 0 이 아닌 값을 반환.

```
int
chain_member_value (tree elem, tree chain)
```

만약 ELEM 이 chain CHAIN 의 어떤 부분인 TREE_VALUE (CHAIN) 와 같다면 0 이 아닌 값을 반환한다. 이것과 다음 함수는 현재로써는 사용되지 않지만, 완료를 위해서 계속 유지된다.

```
int
chain_member_purpose (tree elem, tree chain)
```

만약 ELEM 이 chain CHAIN 의 어떤 부분인 TREE_PURPOSE (CHAIN) 와 같다면 0 이 아닌 값을 반환한다.

```
int
list_length (tree t)
```

TREE_CHAIN 을 통해 연결된 node 들의 chain 길이를 반환한다. 우리는 chain 의 끝이 null pointer 로 마크된 것으로 예상한다. 이것은 Lisp 시대의 'length' 이다.

```
int
fields_length (tree type)
```

TYPE 내의 FIELD_DECL 들의 갯수를 반환한다.

```
tree
chainon (tree op1, tree op2)
     tree op1, op2;
```

chain 1 에서의 마지막 node 를 chain 2 를 가르키게 함으로써 두 chain node 들을 연결합니다. (TREE_CHAIN 으로 연결된 chain 들) 이것은 Lisp 시대의 'nconc' 이다.

```
tree
tree_last (tree chain)
```

Node 들의 chain (TREE_CHAIN 으로 연결된 것) 에서 마지막 node 를 반환합니다.

```
tree
nreverse (tree t)
```

Chain T 내의 요소들의 순서를 반대로 합니다. 그리고 새로운 chain 의 처음 것 (이전의 마지막 요소) 을 반환합니다.

```
tree
listify (tree chain)
```

주어진 tree node 들의 chain CHAIN 으로 node 들의 list 를 만들어 반환한다.

```
tree
build_tree_list (tree parm, tree value)
```

Purpose 와 value 필드의 값이 각각 PARM 과 VALUE 인 새로이 생성된 TREE_LIST node 를 반환합니다.

```
tree
tree_cons (tree purpose, tree value, tree chain)
```

Purpose 와 value 필드의 값이 각각 PARM 과 VALUE 이고 TREE_CHIAN 이 CHAIN 값을 갖는 새로이 생성된 TREE_LIST node 를 반환합니다.

```
tree
size_in_bytes (tree type)
```

Type TYPE 의 object 가 메모리상에 거주할 때 명목상으로 할당한 size 를 반환한다. 값이 바이트 조합으로 측정되었으며, 그것의 data type 은 보통 type size 들에 사용되는 것이다. (그것은 make_signed_type 혹은 make_unsigned_type 에 의해 만들어진 첫번째 type 이다.)

```
HOST_WIDE_INT
int_size_in_bytes (tree type)
     tree type;
```

Wide integer 로 TYPE 의 크기 (바이트로) 를 반환한다. 만약 크기가 변동될 수 있거나 정수보다 클 경우 -1 을 반환한다.

```
tree
bit_position (tree field)
```

Record 의 시작에서 비트 크기로, FIELD 의 bit 위치를 반환한다. 이것은 type bitsizetype 의 tree 이다.

```
HOST_WIDE_INT
int_bit_position (tree field)
```

비슷하지만, 정수로써 반환한다. 그러한 방식으로 표현될 수 없다면 멈춘다. (그것이 signed 값일 수 있기 때문에, int_size_in_byte 가 할 수 있는 것과 같이 -1 을 반환하는 option 을 가지고 있지 않다.

```
tree
byte_position (tree field)
```

Record 의 시작 부분부터 바이트 크기로, FIELD 의 byte 위치를 반환한다. 이것은 type sizetype 의 tree 이다.

```
HOST_WIDE_INT
int_byte_position (tree field)
```

비슷하지만, 정수로써 반환한다. 그러한 방식으로 표현될 수 없다면 멈춘다. (그것이 signed 값일 수 있기 때문에, int_size_in_byte 가 할 수 있는 것과 같이 -1 을 반환하는 option 을 가지고 있지 않다.

```
unsigned int
expr_align (tree t)
```

T 가 가질 수 있는 것으로 알려진 가장 엄격한 alignment 를 비트 크기로 반환한다.

```
tree
array_type_nelts (tree type)
```

TYPE (이 ARRAY_TYPE 인) 의 element 들의 갯수에서 1 을 뺀 수를 tree node 형태로 반환한다. 이것은 top array 의 element 들만 센다.

```
int
staticp (tree arg)
```

만약 arg 가 (static storage 내에서 object 로의 reference 인) static 이라면 0 이 아닌 값을 반환한다. 이것은 C 의미에서의 'staitc' 과 같은 것이 아니다.

```
tree
save_expr (tree expr)
```

적절할 경우, EXPR 주위로 SAVE_EXPR 을 포장한다. 하나의 장소보다 더 많은 곳에서 사용될 수 있는 어떠한 표현식에 이것을 적용할 수 있지만, 단 한번만 평가되어져야 한다.

보통, expand_expr 는 매번 표현식을 재평가한다. save_expr 를 호출하는 것은 expand_expr 가 처음 호출되었으며 평가되고 기록되었다는 것을 나타내는 어떤 것을 생산한다. expand_expr 로의 순차적인 호출은 단지 기록된 값을 재사용하기만 한다.

실제 값을 계산하는 code 를 생산하는 expand_expr 로의 호출은 *컴파일 시* 첫번째 호출에 해당한다. *컴파일 시* 순차적인 호출은 저장된 값을 사용하가 위한 code 를 생성한다. 이것은 올바른 결과를 생산해 내는데, 이 결과는 save_expr 가 평가되어진 다른 장소들에 이르기 전, 첫번째 expand_expr 에 의해 만들어진 명령어들을 통한 흐름을 항상 제어하기 위해 *실행 시* 제공되어진다. 당신은, save_expr 의 호출자, 이것이 그렇게 되도록 반드시 확실하게 만들어줘야 한다.

상수들, 그리고 특정 read-only node 들, 은 그것이 안전한 것이기에 SAVE_EXPR 없이 반환한다. placeholder 들을 포함하는 표현식들은 건딜지 않는다; 이것들이 무엇을 위해 사용되지에 대한 설명을 보려면 tree.def 를 참고하라.

```
tree
unsave_expr (tree expr)
```

독립적으로 여러번 확장된 표현식을 정리한다. 이것은 cleanup 행위에 대해 유용한데, backend 는 다른 장소에서 여러번 이것들을 확장할 수 있다.

```
int
first_rtl_op (enum tree_code code)
```

CODE 를 위한 첫번째 non-tree operand 의 index 를 반환하거나, 만약 모든 것이 tree 일 경우 operand 들의 갯수를 반환한다.

```
void
unsave_expr_1 (tree expr)
```

unsave 되어질 때 필요한 EXPR 로의 어떠한 수정들을 실행한다. EXPR 의 하위 tree 들로의 recurse 를 하지 않는다.

```
static void
unsave_expr_now_r (tree expr)
```

unsave_expr_now 함수를 위한 도움 (helper) 함수.

```
tree
unsave_expr_now (tree expr)
```

Tree 를 그 자리에서 수정하는데, 모든 평가한 오직 하나만 깨끗히 된다. 주어진 EXPR 를 반환한다.

```
int
unsafe_for_reeval (tree expr)
```

만약 EXPR 를 여러번 평가해도 안전하다면 0 을 반환한다.
만약 EXPR 가 나중에 unsave 되는데, 안전하다면 1 을 반환한다.
만약 전혀 안전하지 않다면 2 를 반환한다.

이것은 CALL_EXPR 들과 TARGET_EXPR 들이 표현식 tree 내에 절대 복제된 적이 없으며, 그래서 그들을 unsave 하는 것이 안전하며, context 를 둘러싸는 것 또한 올바를 것이라고 가정한다.

SAVE_EXPR 들은 기본적으로 *오직* 표현식 tree 내에 복제되어 나타나며, 경우에 따라 함수 전체를 가로지러 나타날 때고 있다. 그런 까닭에 만약에 당신이 모든 나타남이 UNSAVE_EXPR 밑에서 나타나는 것을 알고 있다면 SAVE_EXPR 를 unsave 하는 것이 안전하다.

RTL_EXPR 들은 평가하는 동안 그들의 rtl 을 소비한다. 그래서 그들을 unsave 할 가능성은 전혀 없다.

```
int
contains_placeholder_p (tree exp)
```

만약 EXP 이 PLACEHOLDER_EXPR 를 포함하고 있다면 1 을 반환한다; 예를 들어 그것이 record 내의 field 에 따라 의존하는 size 혹은 offset 을 표현할 수 도 있다.

```
int
has_cleanups (tree exp)
```

만약 EXP 가 무엇을 함께 나누는 outer scope 를 위한 cleanup 들을 생산하는 어떠한 표현식을 포함하고 있다면 1 을 반환한다. fold 에서 사용된다.

```
tree
substitute_in_expr (tree exp, tree f, tree r)     tree r;
```

주어진 tree EXP 와 FIELD_DECL F, 대체값 (replacement value) R 은 tree 를 반환하는데, R 로 대체되어지는 PLACEHOLDER_EXPR 내의, 모든 F 로의 reference 들의 occurrence 를 가지는 tree 를 반환한다. 우리는 여기서 EXP 가 단지 산술 연산 표현식들 혹은 단지 arglist 내에서만 발생하는 PLACEHOLDER_EXPR 를 가진 CALL_EXPR 만 포함하고 있음을 가정한다는 사실을 알아야 한다.

```
tree
stabilize_reference (tree ref)
```

reference 를 안정시키며, 그래서 우리는 그것의 operand 들이 한번 이상 평가되는 것을 일으키지 않고 여러번 그것을 사용할 수 있다. 안정된 reference 를 반환한다. 이것은 save_expr 의 수단으로 작동하며, 그래서 save_expr 에 관한 코멘트에서 경고 (caveat) 들을 볼 수 있다.

또한 conversion 표현식들도 허락하는데, 그들의 operand 들은 reference 들이다. 표현식의 어떤 다른 종류는 변화된 것 없이 반환된다.

```
tree
stabilize_reference_1 (tree e)
```

stabilize_reference 의 하위루틴; 이것은 reference 들의 하위 tree 를 위해 호출된다. side-effect 들을 가진 어떠한 표현식은 반드시 그것이 오직 한번만 평가되었음을 확실하게 하기 위해 SAVE_EXPR 내에 놓여야 한다.

우리는 모든 것 주위로 SAVE_EXPR node 들을 놓지 않는데, 임시적으로 매우 간단한 표현식을 할당하는 것이 최적화의 좋은 기회를 놓치는 원인이 될 수 있기 때문이다. 다른 것들로 인해, 상수의 덧셈을 addressing mode 내로 fold 하는 기회를 종종 놓치는데, 예를 들면 “y[i+1] += x;” 이다. 일반적으로, 우리가 그것을 강제하지 않는다면 할당을 되도록 만들지 않는 접근법을 사용하는데 - 부연 하면, 어떤한 non-side effect operator 는 되도록 허락되어야 하고, 그리고 그 cse 는 유용한 것으로 입증되었지만 같은 표현식의 여러번 반복을 연합시키는 몫을 담당하여야 한다.

```
tree
build VPARAMS ((enum tree_code code, tree tt, ...))
```

표현식을 만들어 주는 low-level 건설자들.

Code 가 CODE 이고 data type 이 TYPE, 지정된 인자 ARG 1 과 그에 뒤따르는 인자들을 operand 들로 가지는 표현식을 만듭니다. 표현식들과 reference node 들은 이러한 방법으로 생성될 수 있습니다. Constant 들 그리고 decl 들, type 들, 기타 node 들은 적용될 수 없습니다.

```
tree
build1 (enum tree_code code, tree type, tree node)
```

위와 거의 같지만, 오직 unary operator 들을 위해서만 생성한다. ‘build’ 로의 호출에 대한 공유를 절약한다; RISC machine 들에겐 비용이 비싼 varargs 의 사용을 줄인다.

```
tree
build_nt VPARAMS ((enum tree_code code, ...))
```

TREE_TYPE 을 지정하지 않는 것과 TREE_SIDE_EFFECTS 를 0 으로 남겨둔다는 것을 제외하고는 비슷합니다. Argument 들이 NULL 인 것도 허락을 하는데, 심지어 garbage 의 값들이 문제가 되지 않는다면 garbage 도 허락한다.

```
tree
build_decl (enum tree_code code, tree name, tree type)
```

code 가 CODE 이고 name 이 NAME 이며 data type 은 TYPE 인 DECL_... 을 생성합니다. 우리는 이 node 를 어떤 symbol table 의 sort 내에 넣지 않습니다.

layout_decl 은 decl 의 storage layout 을 설정하는데 사용됩니다. 다른 slot 들은 0 혹은 NULL 포인터로 초기화됩니다.

```
tree
build_block (tree vars, tree tags ATTRIBUTE_UNUSED, tree subblocks,
             tree supercontext, tree chain)
```

BLOCK node 들은 binding contour 들과 declaration 을의 구조체를 나타내는 데 사용된다. 그러면 그러한 contour 들은 exit 되고, 그들의 내용들은 컴파일 되어진다. 이 정보는 debugging info 를 output 하는데 사용된다.

```
tree
build_expr_wfl (tree node, const char *file, int line, int col)
```

EXPR_WITH_FILE_LOCATION 는 표현식 혹은 식별자를 만났었던 정확한 위치의 장소를 유지하는데 사용된다. 그것은 frontend parser 가 하나의 파일 이상을 재귀적으로 다루는 언어에 대해 필요하다. (Java 가 그 중 하나이다.)

```
tree
build_decl_attribute_variant (tree ddecl, tree attribute)
```

그것의 DECL_ATTRIBUTES 가 ATTRIBUTE 인 것을 제외한 DDECL 와 비슷한 선언을 반환한다.

```
tree
build_type_attribute_variant (tree ttype, tree attribute)
```

그것의 TYPE_ATTRIBUTE 가 ATTRIBUTE 인 것을 제외한 TTYPE 와 비슷한 type 을 반환한다.

그러한 수정된 type 들을 기록하는데, 이미 만들어져서 우리가 복제품을 만들 필요가 없다.

```
int
default_comp_type_attributes (type1, type2)
     tree type1 ATTRIBUTE_UNUSED;
     tree type2 ATTRIBUTE_UNUSED;
```

항상 1 을 반환하는 targetm.comp_type_attributes 의 기본 버전.

```
void
default_set_default_type_attributes (type)
     tree type ATTRIBUTE_UNUSED;
```

항상 아무 것도 하지 않는 targetm.set_default_type_attributes 의 기본 버전.

```
void
default_insert_attributes (decl, attr_ptr)
     tree decl ATTRIBUTE_UNUSED;
     tree *attr_ptr ATTRIBUTE_UNUSED;
```

항상 아무 것도 하지 않는 targetm.insert_attributes 기본 버전.

```
bool
default_function_attribute_inlinable_p (fndecl)
     tree fndecl ATTRIBUTE_UNUSED;
```

항상 false 를 반환하는 targetm.function_attribute_inlinable_p 의 기본값.

```
bool
default_ms_bitfield_layout_p (record)
     tree record ATTRIBUTE_UNUSED;
```

항상 false 를 반환하는 targetm.ms_bitfield_layout_p 의 기본값.

```
int
is_attribute_p (const char attr, tree ident)
```

만약 IDENT 가 attribute ATTR 을 위한 올바른 이름이라면 0 이 아닌 값을 반환합니다. 아닐 경우 0 을 반환합니다.

우리는 'text' 와 '__text' 둘 다 시도해 봅니다. ATTR 은 이들 중 하나일 것입니다.

```
tree
lookup_attribute (const char *attr_name, tree list)
```

주어진 attribute 이름과 attributes 의 리스트를 사용해서 만약 attribute 가 리스트의 부분이라면 attribute 의 리스트 element 의 포인터를 반환합니다. 만약 찾지 못했다면 NULL_TREE 를 반환합니다. 만약 attribute 가 한번 이상 나타난다면 이것은 처음 알아낸 것을 반환합니다. – Return 값의 TREE_CHAIN 은 만약 나중에 발생이 필요로 할 경우 되돌려 건네주어야 한다.

```
tree
merge_attributes (tree a1, tree a2)
```

a1 과 a2 의 union 인 attribute list 를 반환한다.

```
tree
merge_type_attributes (tree t1, tree t2)
```

주어진 type 들 T1 과 T2 에서 그들의 attribute 들을 합병하고 그 결과를 반환한다.

```
tree
merge_decl_attributes (tree olddecl, tree newdecl)
```

주어진 decl 들 OLDDECL 과 NEWDECL 에서 그들의 attribute 들을 합병하고 그 결과를 반환한다.

```
tree
merge_dllimport_decl_attributes (tree old, tree new)
```

여러 Windows target 들을 위한 merge_decl_attributes 의 전문 분야.

이것은 다음의 상황을 다룬다:

```
__declspec (dllimport) int foo;
int foo;
```

'foo' 의 두번째 instance 는 dllimport 를 무효로 한다.

```
static void
set_type_quals (tree type, int type_quals)
```

TYPE 을 TYPE_QUALS 로 type qualifier 들을 설정한다. 이것은 여러 TYPE_QUAL 값들의 bitmask 이다.

```
tree
get_qualified_type (tree type, int type_quals)
```

만약 존재할 경우, TYPE_QUALS 에 의해 가르켜지는 것으로 자격이 부여된 TYPE 의 version 을 반환한다. 만약 자격이 부여된 version 이 아직 존재하지 않을 경우, NULL_TREE 를 반환한다.

```
tree
build_qualified_type (tree type, int type_quals)
```

get_qualified_type 와 비슷하지만 만약 type 이 존재하지 않을 경우, 생성한다. 이 함수는 NULL_TREE 를 절대 반환하지 않는다.

```
tree
build_type_copy (type)
     tree type;
```

TYPE 의 새 variant 를 생성하는데, 동일하지만, 구분된다. 이것은 호출자 (caller) 가 그것을 수정 가능하다.

```
unsigned int
type_hash_list (tree list)
```

Type 들의 list (TREE_VALUE slot 들내에 type 들을 가지고 있는 TREE_LIST node 들의 chain) 를 위한 hash code 를 계산한다. 그것은 개별적인 type 들의 hash code 들을 더함으로써 이루어진다.

```
static int
type_hash_eq (const void *va, const void *vb)
```

hashtable callback 함수들이다. 만약 type 들이 같다면 true 를 반환.

```
static unsigned int
type_hash_hash (const void *item)
```

hashtable callback 함수들이다. Cache 된 hash 값을 반환.

```
tree
type_hash_lookup (unsigned int hashcode, tree type)
```

hashtable callback 함수들이다. TYPE 과 같은 모양의 type 에 대해 type hash table 내에서 찾는다. 만약 하나가 찾았다면, 그것을 반환하고, 그렇지 않다면 0 을 반환한다.

```
void
type_hash_add (unsigned int hashcode, tree type)
```

hashtable callback 함수들이다. type TYPE 의 hash code 가 HASHCODE 인 것을 type-hash-table 에 추가한다.

```
tree
type_hash_canon (unsigned int hashcode, tree type)
```

hashtable callback 함수들이다. 주어진 TYPE 과 그것의 hash code 가 HASHCODE 인 것이 이미 존재한다면 그에 해당하는 동일한 type 을 위한 canonical object 를 반환한다. 그렇지 않다면, TYPE 을 반환하고, 만약 그것이 permanent object 일 경우 canonical object 로써 그것을 기록한다.

이 함수를 사용하기 위해, 먼저 당신이 원하는 sort 의 type 을 생성하라 그런후 type 의 field 들로부터 그것의 hash code 를 계산하라. 그 계산은 다른 비슷한 type 들과 다르게 만들 것이다. 그런후 이 함수를 호출하고 해당 값을 사용하면 된다. 이 함수는 만약 당신이 건넨 type 이 중복될 경우 그것을 free 시킬 것이다.

```
static int
type_hash_marked_p (const void *p)
```

Type hash table 로 가르켜지는 data 가 mark 되었는지를 봅니다. 우리는 type 이 이미 mark 되어 있는가, 혹은 debug type number 혹은 symbol table entry 가 type 을 위해 이미 생성되었는가를 고려합니다. 이 방식은 debugging output 의 양을 줄이고 garbage collection 들의 숫자에 의한 debug output 의 의존성을 없앱니다.

```
static void
type_hash_mark (const void *p)
```

가르키는 type 이 mark 되었다고 type hash table 내의 entry 를 mark 합니다. 또한 이 entry 가 TYPE_SYMTAB_POINTER 가 설정됨으로써의 효력에 의해 mark 되었음 나타내는 고려하고 있는 현재 경우의 type 을 mark 합니다.

```
static int
mark_tree_hashtable_entry (void **entry, void *data ATTRIBUTE_UNUSED)
```

GC 용 ENTRY (는 실제로 'tree**' 이다) 에 의해 가르켜지는 hashtable slot 를 mark 한다.

```
void
mark_tree_hashtable (void *arg)
     void *arg;
```

GC 용 ARG (는 실제 htab_t 인데, htab_t 의 slot 들은 tree 이다.) 를 mark 한다.

```
static void
print_type_hash_statistics ()
```

이름에서 알수 있듯이, Type hash 통계를 출력한다.

```
unsigned int
attribute_hash_list (tree list)
```

attribute 들의 list 를 위한 hash code 를 계산하는데, (TREE_PURPOSE slot 들내에 이름들을 가지고, TREE_VALUE slot 들내에 arg 들을 가지는 TREE_LIST node 들의 chain) 개별적인 attribute 들의 hash code 들을 더함으로써 이루어 진다.

```
int
attribute_list_equal (tree l1, tree l2)
```

주어진 attribute 들의 두 list 를 이용하여 만약 list l2 가 l1 과 동일하다면 true 를 반환한다.

```
int
attribute_list_contained (tree l1, tree l2)
```

주어진 attribute 들의 두 list 를 이용하면 만약 list L2 가 L1 내에 완전히 포함된다면 true 를 반환한다.

```
int
type_list_equal (tree l1, tree l2)
```

주어진 type 들의 두 list 들을 이용하여, (TREE_VALUE slot 들내에 type 들 가지는 TREE_LIST node 들의 chain 들), 만약 list 들이 같은 순서로 같은 type 들을 포함하고 있다면 1 을 반환한다. TREE_PURPOSE 들도 반드시 맞아 떨어져야 한다.

```
int
type_num_arguments (tree type)
     tree type;
```

TYPE 에 의해 주어진 FUNCTION_TYPE 혹은 METHOD_TYPE 에서 argument 들의 갯수를 반환한다. 만약 argument list 가 variable argument 들을 받아 들인다면, 이 함수는 보통의 argument 들만을 센다.

```
int
tree_int_cst_equal (tree t1, tree t2)
```

만약 정수 상수들 T1 과 T2 가 같은 상수를 나타낸다면 0 이 아닌 값.

```
int
tree_int_cst_lt (tree t1, tree t2)
```

만약 정수 상수들 T1 과 T2 가 ¡ 를 만족하는 값들을 나타낸다면 0 이 아닌 값. 비교하는데 있어서 접근 방법은 그들의 data type 에 의존한다.

```
int
tree_int_cst_compare (tree t1, tree t2)
```

만약 T1 < T2 이면 -1 을, T1 == T2 이면 0 을, T1 > T2 이면 1 을 반환한다.

```
int
host_integerp (tree t, int pos)
```

만약 T 가 호스트상에서 효율적으로 다루어 질 수 있는 INTEGER_CST 라면 1 을 반환한다. 만약 POS 가 0 이라면, 값은 단일 HOST_WIDE_INT 로 나타날 수 있다. 만약 POS 가 0 이 아니라면, 값은 반드시 양수여야 하고, 단일 unsigned HOST_WIDE_INT 내로 나타낼 수 있어야 한다.

```
HOST_WIDE_INT
tree_low_cst (tree t, int pos)
```

만약 INTEGER_CST 이고 overflow 가 없다면 T 를 위한 약간 충분한 bit 들을 가지는 HOST_WIDE_INT 를 반환한다. 만약 결과가 반드시 양수여야 한다면 POS 가 0 이 아니다. 만약 우리가 위의 조건을 만족할 수 없다면 멈춘다.

```
int
tree_int_cst_msb (tree t)
```

정수 상수 T 를 위한 가장 충분히 큰 bit 를 반환한다.

```
int
tree_int_cst_sgn (tree t)
```

정수 상수 T 의 부호에 관해 반환한다.
반환값은 T < 0 이면 -1, T == 0 이면 0, T > 0 이면 1 이다. T 의 type 이 unsign 일 때 절대 -1 이 반환되지 않음을 참고하라.

```
int
simple_cst_list_equal (tree l1, tree l2)
```

두개의 constructor-element-type 상수들을 비교한다. 만약 list 들이 같을 경우 1 을 반환; 그렇지 않을 경우 0 을 반환한다.

```
int
simple_cst_equal (tree t1, tree t2)
```

T1 이 T2 와 같은 tree 구조체인지에 상관없이 truthvalue 를 반환한다.
만약 이것들이 같으면 1 을 반환.
만약 이것들이 이해가 갈 정도로 다르면 0 을 반환.
만약 tree 구조체를 포함하지도 않고 이 함수에 의해 이해되지도 않는다면 -1 을 반환한다.

```
int
compare_tree_int (tree t, unsigned HOST_WIDE_INT u)
```

INTEGER_CST 인 T 의 값과 unsigned 정수값인 U 의 값을 비교한다. 만약 T 의 값이 U 보다 작거나, 같거나, 크다면, 그에 따라 -1 혹은 0, 1 을 반환한다.

```
tree
build_pointer_type (tree to_type)
```

Pointer 와 array, function type 들을 위한 함수들. (RECORD_TYPE 와 UNION_TYPE, ENUMERAL_TYPE node 들은 language-dependent code 에 의해 생성되며, 여기서는 아니다.)

TO_TYPE 로의 pointer 들의 type 을 만들어, 레이아웃해서 반환한다. 만약 그러한 type 이 이미 만들어 졌다면, 그것을 재사용한다.

```
tree
build_reference_type (to_type)
     tree to_type;
```

references-to-TO_TYPE 의 type 을 위한 node 를 생성한다.

```
tree
build_type_no_quals (t)
     tree t;
```

t 와 호환성있는 type 을 생성하지만, 그것의 type 내의 어느 곳에도 cv qual 들을 가지고 있지 않는다, 그래서

```
   const char *const *const *  ->  char ***
```

와 같이 된다.

```
tree
build_index_type (maxval)
     tree maxval;
```

ARRAY_TYPE 의 TYPE_DOMAIN 를 위한 정수들의 type 을 생성한다. MAXVAL 는 domain 내에서의 최대값이여야한다. (각각은 배열의 길이보다는 작아야 한다.)

MAXVAL 가 가질 수 있는 최대값은 HOST_WIDE_INT 에서의 INT_MAX 이다. 우리는 이 한계를 강제로 정하지 않는데, 그것은 호출자의 책임이다. (예를 들면 , 언어의 front end) 한계는 결과가 signed type 이고 하나의 HOST_WIDE_INT 보다 더 큰 사용을 하는 size 들은 우리가 다루지 않기 때문에 존재한다.

```
tree
build_range_type (type, lowval, highval)
     tree type, lowval, highval;
```

low bound 가 LOWVAL 이고, high bound 가 HIGHVAL 를 가지는 몇몇 discrete type TYPE (CHAR_TYPE 혹은 INTEGER_TYPE, ENUMERAL_TYPE, BOOLEAN_TYPE) 의 범위 (range) 를 생성한다. 만약 TYPE==NULL_TREE 이면, sizetype 이 사용된다.

```
tree
build_index_2_type (lowval, highval)
     tree lowval, highval;
```

build_index_type 와 거의 같지만, 단순히 highval (maxval) 대신에 lowval 와 highval 를 가진다.

```
int
index_type_equal (itype1, itype2)
     tree itype1, itype2;
```

만약 ITYPE1 와 ITYPE2 가 같다면 (LISP sense 에서) 0 이 아닌 값을 반환한다. 이것은 index type 들이 hash 되지 않을 경우와 equal index type 들이 비록 체계적이지만, 분명 다른 시기때 나타나서 서로 연관성이 없이 생성되었을 경우 때문에 필요하다.

```
tree
build_array_type (elt_type, index_type)
     tree elt_type, index_type;
```

ELT_TYPE 를 가지고, INDEX_TYPE 의 값의 범위로 지정된 element 들의 갯수 를 가지는 element 들의 array 들의 type 을 생성하고, lay out 한 후 반환한다. 만약 그러한 type 이 이미 생성되었었다면, 그것을 재사용한다.

```
tree
get_inner_array_type (array)
     tree array;
```

ARRAY 의 innermost dimension 을 구성하는 element 들의 TYPE 을 반환한다.

```
tree
build_function_type (value_type, arg_types)
     tree value_type, arg_types;
```

type 들 ARG_TYPES 의 argument 들로 주어진 type VALUE_TYPE 를 반환하는 함수들의 type 을 생성하고, layout 한 후 반환한다. ARG_TYPES 는 TREE_LIST node 들의 chain 인데, TREE_LIST node 의 TREE_VALUE 들이 함수의 argument 들을 위한 data type node 들임을 나타낸다. 만약 그러한 type 이 이미 구성되어 있다면 그것을 재사용합니다.

```
tree
build_method_type (basetype, type)
     tree basetype, type;
```

class BASETYPE 에 속하고 그것의 argument 들과 값들이 TYPE 으로 표현되는 method 들의 type 을 생성하고, layout 한 후 반환한다. 만약 그 type 이 이미 존재한다면, 그것을 재 사용한다. TYPE 은 반드시 FUNCTION_TYPE node 여야 한다.

```
tree
build_offset_type (basetype, type)
     tree basetype, type;
```

type BASETYPE 의 ojbect 내에 존재하면서, type TYPE 의 값인 offset 들의 type 을 생성하고 layout 한 후 반환한다. 만약 적당한 offset type 이 이미 존재할 경우, 그것을 재사용한다.

```
tree
build_complex_type (tree component_type)
```

Component 들이 COMPONENT_TYPE 인 complex type 을 생성한다.

```
tree
get_unwidened (op, for_type)
     tree op;
     tree for_type;
```

안전한 만큼의 wider type 들로의 어떠한 변환들을 잘라낸, OP 를 반환한다. OP 의 type 으로 되돌리는 값의 변환은 값을 OP 와 같게 만든다.

만약 FOR_TYPE 이 0 이 아니라면, 우리는, type FOR_TYPE 으로 변환되었다면, OP 를 type FOR_TYPE 으로 변환한 것과 같은 값을 반환할 것이다.

만약 FOR_TYPE 가 0 이 아니라면, unaligned bit-field reference 들은, 그것이 알맞지 않을 수 있지만, 값을 잡고 있을 수 있는 가장 작은 type 으로 변경 될 수 있다. 그렇지 않을 경우, bit-field reference 들은 해당 type 내 memory 로부터 직접적으로 fetch 될 수 있을 경우에만 좀 더 작은 type 으로 변경될 것이다.

OP 는 반드시 정수, 실수, 혹은 enumeral type 를 가져야 한다. 포인터들은 허락하지 않는다.

우리가 반환할 수 있는 분명한 값이 OP 의 type 이 변환될 경우 OP 로 재 생성될 수 있는 몇몇 경우가 존재한다. 하지만 좀 더 넓은 type 으로 OP 를 확장하지는 않는다. 만약 FOR_TYPE 가 그러한 확장에 관해 심사숙고 하였음을 가르킨다면, 우리는 그러한 값들을 의도적으로 삼가한다. 예를 들어, 만약 OP 가 (unsigned short)(signed char)-1 라면, 우리는 만약 FOR_TYPE 이 int 이고, 그것을 unsigned short 로 확장하는 것이 OP 를 재생성하는 것이라면 (signed char)-1 를 반환하는 것을 피한다. 그렇기 때문에, (signed char)-1 를 (int) 로 확장한 결과는 (int) OP 와는 다르다.

```
tree
get_narrower (tree op, int *unsignedp_ptr)
```

OP 혹은 좀 더 좁은 값을 위한 간단한 표현식을 반환한다. 여기서 더 좁은 값은 OP 를 되돌려 주기 위해 sign-extend 되거나 혹은 zero-extend 되어 질 수 있다. 만약 값이 반드시 zero-extend 되어야 한다면 1, 만약 값이 sign-extend 되어야 한다면 0 을 무조건 *UNSIGNEDP_PTR 에 저장한다.

```
int
int_fits_type_p (tree c, tree type)
```

만약 정수 상수 C 가 type TYPE (INTEGER_TYPE 인) 에 대해 허용할 수 있는 값을 가지고 있다면 0 이 아닌 값을 반환한다.

```
tree
get_containing_scope (tree t)
```

주어진 DECL 혹은 TYPE 을 이용하여, 선언되었던 scope 를 반환하며, 만약 scope 를 포함하고 있지 않다면 NULL_TREE 를 반환한다.

```
tree
decl_function_context (tree decl)
```

FUNCTION_DECL 이거나, 존재하지 않을 경우 0 인 DECL 을 동봉하는 innermost context 를 반환한다.

```
tree
decl_type_context (tree decl)
```

RECORD_TYPE 혹은 UNION_TYPE, QUAL_UNION_TYPE 혹은 존재하지 않을 경우 0 인 DECL 를 동봉하는 innermost context 를 반환한다. TYPE_DECL 들과 FUNCTION_DECL 들은 이 함수에 대해 투명하다. (투과성이 있다.)

```
tree
get_callee_fndecl (tree call)
```

CALL 은 CALL_EXPR 이다. 호출된 함수를 위한 declaration 을 반환하거나 만약 호출된 함수가 결정될 수 없을 경우 NULL_TREE 를 반환한다.

```
void
print_obstack_statistics (const char *str, struct obstack *o)
```

Obstack 0 이고, 이름이 STR 인 것에 대한 디버깅 정보를 출력한다.

```
void
dump_tree_statistics ()
```

컴파일 동안에 생성된 tree node 들에 관한 디버깅 정보와 language-specific 정보들을 출력한다.

```
static void
append_random_chars (char *template)
```

우리가 믿음직한 유일한 이름을 고를수 없는 경우를 (제발) 피하기 위해 TEMPLATE 에 6 개의 임의의 문자들을 첨부한다.

libiberty 내의 mkstemp.c 에서 추출하였다.

```
void
clean_symbol_name (char *p)
```

P 는 symbol 내에서 사용될 문자열이다. 현 context 내에 유효한 것이 아닌 문자에 가면을 쒸웁니다. (문자를 다른 것으로 대체.)

```
tree
get_file_function_name_long (const char *type)
```

이 translation unit 에 대해 함수를 위한 유일한 이름을 생성한다. TYPE 은 linker 혹은 collect2 에 대해 이 함수의 목적을 확인할 수 있도록 도와주는 몇몇 문자열이다.

```
tree
get_file_function_name (int kind)
```

만약 KIND==‘I’ 이면, 적당한 global initializer (constructor) name 을 반환한다.
만약 KIND==‘D’ 이면, 적당한 global clean-up (destructor) name 을 반환한다.

```
tree
get_set_constructor_bits (tree init, char *buffer, int bit_size)
```

SET_TYPE CONSTRUCTOR node (의 상수 부분) 를 확장한다. 결과는 (길이가 BIT_SIZE 인) BUFFER 에 놓이게 되며, 각 char 내 한 bit 만 (‘\000’ 혹은 ‘\001’) 가진다.

만약 constructor 가 상수이라면, NULL_TREE 가 반환된다. 그렇지 않을 경우, 비-상수 요소의 TREE_LIST 가 나오게 된다.

```
tree
get_set_constructor_bytes (tree init, unsigned char *buffer, int wd_size)
```

SET_TYPE CONSTRUCTOR node (의 상수 부분) 를 확장한다. 결과는 (byte 들의 배열인) BUFFER 에 놓이게 된다. 만약 constructor 가 상수이라면, NULL_TREE 가 반환된다. 그렇지 않을 경우, 비-상수 요소의 TREE_LIST 가 나오게 된다.

```
void
tree_check_failed (node, code, file, line, function)
     const tree node;
     enum tree_code code;
     const char *file;
     int line;
     const char *function;
```

NODE 의 tree code 가 예상된 CODE 와 맞지 않다고 불평한다. FILE 과 LINE, FUNCTION 은 호출자의 것이다.

```
void
tree_class_check_failed (node, cl, file, line, function)
     const tree node;
     int cl;
     const char *file;
     int line;
     const char *function;
```

위와 비슷하지만, 우리는 CL 내에 주어진 tree code 의 class 를 비교한다는 것이 다르다.

```
static void
finish_vector_type (tree t)
```

새로운 vector type node T 를 위해, output 을 디버깅할 때 필요한 정보를 생성한다.

```
void
build_common_tree_nodes (int signed_char)
```

C datatype 들의 크기에 사용되는 모든 정수형 type (error_mark_node 도 포함) 을 위한 node 들을 생성합니다. 호출자는 sizetype 으로 type 들 중 하나를 선택해야 하기 때문에 이 함수를 호출한 후 set_sizetype 을 곧바로 호출해 줘야 합니다.

```
void
build_common_tree_nodes_2 (int short_double)
```

build_common_tree_nodes 와 set_sizetype 를 호출한 후 이 함수를 호출한다. 이것은 여러 다른 common tree node 들을 생성할 것이다.

```
static tree
make_vector (enum machine_mode mode, tree innertype, int unsignedp)
```

주어진 vector mode 와 내부(inner) type, 부호 를 가지는 vector tree 를 반환한다.

제 8 절 14 차 강의를 마치며

휴~ 아주 많은 시간을 끌게 만든 TREE 에 대한 정리가 큰 부분에 대해서만 이제 끝난 듯합니다. 실제 작동 원리에 대해서는 구체적으로 적지는 못하였지만, 포괄적인 부분에 대한 수용은 이루어졌다고 봅니다. 작은 산은 정복하였지만 아직도 갈 길이 너무나 많이 남아 있네요. 다른 부분들은 다음 강의에서 봅시다. 그럼.