Next
Wednesday,
03rd February 2010
Article in Spanish

Exporting and Visualizing Gcc's Abstract Syntax Tree (AST)

An AST is a program representation after being parsed by a compiler or an interpreter.

Here we can see the steps to visually represent this tree. In this case, we will make use of Gcc compiler.

We will need:

- Gcc
- Gawk
- Graphviz

First of all we create a file named *test.c* with the familiar Hello World program.

test.c

```
#include<stdio.h>
int main(int arg_count,char ** arg_values)
{
    printf("Hello World\n");
    return 0;
}
```

We execute this command to pass the correct parameters to Gcc.

ast-raw.sh

```
gcc -fdump-tree-original-raw ./test.c
```

It will generate this other file with the tree dump.

test.c.003t.original

```
;; Function main (null)
;; enabled by -tree-original
@1      bind_expr        type: @2       body: @3
@2      void_type        name: @4       algn: 8
@3      statement_list   0   : @5       1   : @6
@4      type_decl        name: @7       type: @2       srcp: <built-in>:0
@5      call_expr        type: @8       fn  : @9       0   : @10
@6      return_expr      type: @2       expr: @11
@7      identifier_node  strg: void     lngt: 4
@8      integer_type     name: @12      size: @13      algn: 32
```

```
                        prec: 32       sign: signed   min : @14
                        max : @15
@9      addr_expr       type: @16      op 0: @17
@10     nop_expr        type: @18      op 0: @19
@11     modify_expr     type: @8       op 0: @20      op 1: @21
@12     type_decl       name: @22      type: @8       srcp: <built-in>:0
@13     integer_cst     type: @23      low : 32
@14     integer_cst     type: @8       high: -1       low : -2147483648
@15     integer_cst     type: @8       low : 2147483647
@16     pointer_type    size: @24      algn: 64       ptd : @25
@17     function_decl   name: @26      type: @25      srcp: stdio.h:339
                        body: undefined               link: extern
@18     pointer_type    qual:   r      unql: @27      size: @24
                        algn: 64       ptd : @28
@19     addr_expr       type: @29      op 0: @30
@20     result_decl     type: @8       scpe: @31      srcp: test.c:4
                        note: artificial              size: @13
                        algn: 32
@21     integer_cst     type: @8       low : 0
@22     identifier_node strg: int      lngt: 3
@23     integer_type    name: @32      size: @24      algn: 64
                        prec: 64       sign: unsigned min : @33
                        max : @34
@24     integer_cst     type: @23      low : 64
@25     function_type   size: @35      algn: 8        retn: @8
                        prms: @36
@26     identifier_node strg: printf   lngt: 6
@27     pointer_type    size: @24      algn: 64       ptd : @28
@28     integer_type    qual: c        name: @37      unql: @38
                        size: @35      algn: 8        prec: 8
                        sign: signed   min : @39      max : @40
@29     pointer_type    size: @24      algn: 64       ptd : @41
@30     string_cst      type: @41     strg: Hello World
    gt: 13
@31     function_decl   name: @42      type: @43      srcp: test.c:3
                        args: @44      link: extern
@32     identifier_node strg: bit_size_type           lngt: 13
@33     integer_cst     type: @23      low : 0
@34     integer_cst     type: @23      low : -1
@35     integer_cst     type: @23      low : 8
@36     tree_list       valu: @18
@37     type_decl       name: @45      type: @38      srcp: <built-in>:0
@38     integer_type    name: @37      size: @35      algn: 8
                        prec: 8        sign: signed   min : @39
                        max : @40
@39     integer_cst     type: @38      high: -1       low : -128
@40     integer_cst     type: @38      low : 127
@41     array_type      size: @46      algn: 8        elts: @38
                        domn: @47
@42     identifier_node strg: main     lngt: 4
@43     function_type   size: @35      algn: 8        retn: @8
                        prms: @48
@44     parm_decl       name: @49      type: @8       scpe: @31
                        srcp: test.c:3                argt: @8
                        size: @13      algn: 32       used: 0
@45     identifier_node strg: char     lngt: 4
@46     integer_cst     type: @23      low : 104
@47     integer_type    size: @24      algn: 64       prec: 64
                        sign: signed   min : @50      max : @51
@48     tree_list       valu: @8       chan: @52
@49     identifier_node strg: arg_count               lngt: 9
@50     integer_cst     type: @53      low : 0
@51     integer_cst     type: @53      low : 12
@52     tree_list       valu: @54      chan: @55
```

```
@53     integer_type     name: @56      size: @24      algn: 64
                         prec: 64       sign: unsigned min : @57
                         max : @58
@54     pointer_type     size: @24      algn: 64       ptd : @59
@55     tree_list        valu: @2
@56     type_decl        name: @60      type: @61      srcp: <built-in>:0
@57     integer_cst      type: @61      low : 0
@58     integer_cst      type: @53      high: -1       low : -1
@59     pointer_type     size: @24      algn: 64       ptd : @38
@60     identifier_node  strg: long unsigned int       lngt: 17
@61     integer_type     name: @56      size: @24      algn: 64
                         prec: 64       sign: unsigned min : @57
                         max : @62
@62     integer_cst      type: @61      low : -1
```

Next, we will need the following files:

pre.awk

```
#! /usr/bin/gawk -f
/^[^;]/{
    gsub(/^@/, "~@", $0);
    gsub(/( *):( *)/, ":", $0);
    print;
}
```

treeviz.awk

```
#! /usr/bin/gawk -f
#http://alohakun.blog7.fc2.com/?mode=m&no=355
BEGIN {RS = "~@"; printf "digraph G {\n node [shape = record];";}
/^[0-9]/{
s = sprintf("%s [label = \"{%s | {", $1, $1);
for(i = 2; i < NF - 1; i++)
    s = s sprintf("%s | ", $i);
    s = s sprintf("%s}}\"];\n", $i);
    $0 = s;
    while (/([a-zA-Z]+):@([0-9]+)/){
        format = sprintf("\\1 \\3\n %s:\\1 -> \\2;", $1);
        $0 = gensub(/([a-zA-Z]+):@([0-9]+)(.*)$/, format, "g");
    };
    printf " %s\n", $0;
}
END {print "}"}
```

We almost have it. We execute the following:

ast2dot.sh

```
./pre.awk test.c.* | ./treeviz.awk > tree.dot
```

Finally, we use dot (included as part of Graphviz) to create the image:

dot2png.sh

```
dot -Tpng tree.dot -o tree.png
```

There are a few mistakes, here is the code improved:

```
#! /usr/bin/gawk -f
#http://alohakun.blog7.fc2.c...
BEGIN {RS = "~@"; printf "digraph G {\n node [shape = record];";}
/^[0-9]/{
s = sprintf("\n%s [label = \"{%s | {", $1, $1);
for(i = 2; i < NF; i++)
s = s sprintf("%s | ", $i);
s = s sprintf("%s}}\"];", $i);
$0 = s;
while (/([a-zA-Z0-9]+):@([0-9]+)/){
format = sprintf("\\1 \\3\n %s:\\1 -> \\2;", $1);
$0 = gensub(/([a-zA-Z0-9]+):@([0-9]+)(.*)$/, format, "g");
};
printf " %s", $0;
}
END {print "\n}"}
```