Compiladores/Projecto de Compiladores/Perguntas e Respostas sobre o Desenvolvimento (FAQ)

< Compiladores | Projecto de Compiladores

AVISOS - Avaliação em Época Normal [Expand]

Material de Uso Obrigatório [Expand]

Contents [hide]

- 1 Problemas Gerais de Compilação
  - 1.1 Bibliotecas e Interfaces
  - 1.2 Compilação e localização de "expression_type"
  - 1.3 Compilação e "undefined referente to vtable"
- 2 Linguagem "Simple"
  - 2.1 Compilação de Programas "Simple"
  - 2.2 Compilação de um programa Simple para Produzir outra Saída
  - 2.3 Compilação do Simple: doxygen
  - 2.4 Produção de Executáveis (incompatibilidades)
  - 2.5 Produção de um Executável
  - 2.6 Gramática do Simple: %prec, %nonassoc, etc.
  - 2.7 Gramática do Simple: token UMINUS

Problemas Gerais de Compilação

Bibliotecas e Interfaces

Q: Se tiver uma biblioteca **libteste.a** compilada a partir de **a.h** e de **b.h** e um programa que a usa fazendo #include de **a.h** e de **b.h**, não consigo compilar o programa sem ter os **.h** originais?
R: Se o programa cliente usa os nomes **a.h** e **b.h**, então eles têm de existir (mesmo que não sejam os originais). Em geral, a interface (**.h**) de uma biblioteca é independente do código utilizado na sua produção: podem existir **.h** (C/C++) para bibliotecas produzidas a partir de Fortran, por exemplo. Um exemplo concreto é a interface para o standard I/O do C: em C está acessível via **stdio.h** e em C++ via **cstdio**.

Compilação e localização de "expression_type"

Q: Ao executar "make" para construir o projecto, estou a ter o seguinte erro:

```
/usr/include/cdk/ast/expression_node.h:47:38: fatal error: target/expression_type.h: No such file or directory
compilation terminated.
make: *** [zu_parser.tab.o] Error 1
```

Que se passa?
R: O problema ocorre devido ao valor incorrecto de uma variável no ficheiro **Makefile** do projecto. O valor da variável **ROOT** deve ser definido como se segue:

```
ROOT =
```

(ou seja, a variável tem de ser definida sem qualquer valor)

Compilação e "undefined referente to vtable"

Q: Estou a compilar o projecto e ocorre o erro: "undefined referente to vtable" na class basic_ast_visitor. Que se está a passar?
R: O erro ocorre no projecto, geralmente na hierarquia dos "visitors", devido a não terem sido definidos todos os métodos declarados nos "visitors". Note-se que não basta declará-los no cabeçalho da classe: é necessário definir os seus corpos, mesmo que sejam vazios.

Linguagem "Simple"

Compilação de Programas "Simple"

O nome do compilador corresponde sempre ao nome da linguagem. O processo envolve três etapas: compilação do código fonte, compilação do código gerado (pelo yasm/nasm) e ligação dos módulos binários no executável final (incluindo a **librts.a**).

Assumindo a distribuição do Simple na directoria actual e a directoria "examples" com os programas de teste, as seguintes acções seriam necessárias para compilar o exemplo 1 (**ex1.spl**):

```
./simple examples/ex1.spl
yasm -felf32 examples/ex1.asm
ld -o examples/ex1 examples/ex1.o -lrts
```

Os vários comandos podem diferir nos parâmetros exactos (dependendo do ambiente). Por exemplo, se a **librts.a** não estiver numa localização conhecida pelo **ld**, então é necessário dar o comando da seguinte forma:

```
ld -m elf_i386 -o examples/ex1 examples/ex1.o -L$HOME/compiladores/root/usr/lib -lrts
```

Compilação de um programa Simple para Produzir outra Saída

O nome do compilador corresponde sempre ao nome da linguagem. O processo de produção (tradução) de uma outra linguagem (C, XML, etc.) a partir do original Compact é semelhante ao descrito para a compilação para produção de um executável (nesse caso, a tradução é de Compact para ASM).

Assumindo a distribuição do Compact na directoria actual e a directoria "examples"com os programas de teste, as seguintes acções seriam necessárias para traduzir o exemplo 1 (ex1.spl) para XML:

```
./simple examples/ex1.spl -target xml
```

Neste caso, a saída do compilador é o ficheiro ex1.xml. Resultados semelhantes seriam obtidos para os outros "targets". Um comando equivalente ao anterior é o seguinte:

```
./simple examples/ex1.spl -o examples/ex1.xml
```

Compilação do Simple: doxygen

Q: Quando tento compilar o compilador Simple, o make produz um erro ao tentar executar o programa **doxygen**. O que estou a fazer mal?

R: O programa **doxygen** é utilizado para gerar documentação automaticamente a partir do código (à la Javadoc). É um programa livremente distribuído e está contido em muitas distribuições Linux, assim como para outras plataformas.

Produção de Executáveis (incompatibilidades)

Q: Ao executar o "ld" com os ficheiros criados com o compilador e usando as opções **-L$HOME/compiladores/root/usr/lib -lrts** aparece a seguinte mensagem de erro:

```
ld: skipping incompatible .../compiladores/root/usr/lib/librts.a when searching for -lrts
ld: cannot find -lrts
```

R: Este erro ocorre devido a incompatibilidade entre formatos binários (entre os módulos do programa e a biblioteca), tipicamente, entre 32 bits e 64 bits. A solução é utilizar apenas uma arquitectura ou passar ao "ld" as opções de compilação em 32 bits (ver acima).

Produção de um Executável

Q: Na produção de um executável Simple, depois de realizadas as fases de compilação e geração de módulo binário (.o), devo produzir o executável com o comando **gcc examples/ex1.o -o ex1**?

R: O GCC, tal como invocado acima, só serve para produzir executáveis baseados em C, já que o comando não é mais que uma chamada ao ld em que é passada a libc como argumento:

```
ld -o examples/ex1.o -o ex1 -lc
```

No caso do Simple (ou de outro compilador semelhante), o que se deve fazer é (note-se que é necessário especificar **elf_i386**, o único formato suportado pela RTS):

```
ld -m elf_i386 -o examples/ex1.o -o ex1 -lrts
```

Gramática do Simple: %prec, %nonassoc, etc.

Q: Na especificação da gramática do Simple, o que é o %prec? E o que quer dizer %nonassoc IFX (no .l não é reconhecido nenhum token chamado IFX)?

R: %prec é uma definição de precedência de operadores; %nonassoc indica que um operador não é associativo. Estas directivas permitem ao parser tomar decisões relativamente a situações em que a gramática seria ambígua. **Quando a gramática não é ambígua, as directivas são simplesmente ignoradas**. IFX é um token utilizado pelo parser para resolução de uma potencial ambiguidade na gramática do Compact.

Gramática do Simple: token UMINUS

Q: O que é? Não é detectado pelo Flex.

R: O token UMINUS é um token definido na especificação da gramática do Simple. É utilizado para definir a precedência dos operadores unários representados pelos tokens '-' e '+' (esses, sim, reconhecidos pelo analisador lexical).

Categories: **Projecto de Compiladores** **Compiladores** **Ensino**