MAC–IME–USP

Carlos Eduardo Ferreira
Sala 108C Tel.: 3091 6135
e-mail cef@ime.usp.br
monitor Bruno Klava

MAC 323 – Estruturas de Dados

Primeiro semestre de 2005

Exercício-Programa 2 – Entrega: **3 de maio de 2005**

Make

O objetivo deste exercício-programa é construir um programa que funcione de forma semelhante ao utilitário `make`. O seu programa deverá ser chamado da linha de comando e lerá um arquivo de nome `MakeFile` contendo informações de dependências e comandos para reconstrução de objetos (note que o arquivo a ser lido tem um "F" maiúsculo, a fim de diferenciar do `Makefile` que você possivelmente estará usando).

Formato do `MakeFile`

Um `MakeFile` consiste de uma seqüência de regras com o seguinte formato:

```
<target> : <dependências>
   <comando 1>
   <comando 2>
   ...
```

O `target` é o nome de um arquivo que é gerado por um programa (por ex. um executável) ou uma ação.

A lista de arquivos dos quais depende o `target` é dado em `dependências`, e a construção do `target` é feita através da lista de `comandos`. Antes de cada `comando` existe um caractere TAB.

Exemplo:

```
prog: prog.o prog1.o prog2.o
      gcc prog.o prog1.o prog2.o -o prog

prog.o: prog.c meusdefs.h fila.h pilha.h
      gcc -c prog.c

prog1.o: prog1.c meusdefs.h fila.h
      gcc -c prog1.c

prog2.o: prog2.c meusdefs.h pilha.h
      gcc -c prog2.c

limpa:
      rm prog.o prog1.o prog2.o
```

Note que o objeto `limpa` não depende de nenhum arquivo. O objeto `prog` por sua vez depende de três arquivos. Seu programa deverá ler um arquivo `MakeFile` como descrito acima e, inicialmente, verificar se suas regras são ou não consistentes.

Dizemos que um `MakeFile` tem um conjunto inconsistente de regras se a execução de um de seus objetos for impossível em virtude de suas dependências formarem um ciclo.

Exemplo:

```
prog: prog1.c prog2.o
       gcc prog2.o prog1.c -o prog

prog2.o: prog2.c prog1.o
       gcc -c prog1.o prog2.c

prog1.o: prog1.c prog2.o
       gcc -c prog2.o prog1.c
```

Depois de determinar se o arquivo é consistente (não contém ciclos como o acima, de qualquer comprimento), seu programa deverá imprimir em um arquivo de saída os comandos necessários para a construção do objeto solicitado.

Exemplo: Ao criar o objeto `prog` no exemplo sem ciclos acima, seu programa deverá gerar os targets `prog.o`, `prog1.o` e `prog2.o`, e, então, gera o objeto `prog`. Neste caso, deverá imprimir no arquivo de saída:

```
gcc -c prog.c
gcc -c prog1.c
gcc -c prog2.c
gcc prog.o prog1.o prog2.o -o prog
```

Se o arquivo `prog1.o` estiver atualizado (ou seja, não houve alteração desde a última compilação), por exemplo, seu programa deverá gerar apenas os targets `prog.o` e `prog2.o` antes de gerar o objeto `prog`, imprimindo:

```
gcc -c prog.c
gcc -c prog2.c
gcc prog.o prog1.o prog2.o -o prog
```

Observe que se apenas o arquivo `prog2.c` for alterado, ao solicitar a construção do objeto `prog`, seu programa deverá gerar `prog2.o` e em seguida gerar o objeto `prog`, imprimindo no arquivo de saída:

```
gcc -c prog2.c
gcc prog.o prog1.o prog2.o -o prog
```

A estrutura de dados

Ao ler o arquivo `MakeFile`, seu programa deve montar uma estrutura de dados usando listas duplamente ligadas circulares com cabeça de lista para armazenar as dependências. Cada `target` deverá apontar para uma lista ligada com os arquivos de que depende. Deverá haver também uma tabela de targets. Na figura abaixo mostramos a estrutura correspondente ao `MakeFile` do exemplo acima.

Tabela de targets