Universidade Federal Fluminense
Escola de Engenharia
Curso de Engenharia de Telecomunicações
Programa de Educação Tutorial

Dicas PET-Tele

Noções básicas sobre MAKE

Autor: Mathyan Motta Beppu
Tutor: Alexandre Santos de la Vega

Niterói-RJ
Julho / 2010

1 Definição

O *makefile* é um arquivo de configuração de compilações, este arquivo é utilizado pelo programa *make*, cujo o principal objetivo é otimizar, simplificar e agilizar a compilação de programas.

2 Benefícios de sua utilização

O programa *make* possui vários benefícios, como, por exemplo:

- Otimiza a compilação de programas, evitando a recompilação de arquivos desnecessários: suponha que você esteja compilando um programa que utiliza 10 arquivos e você altera apenas um desses, o *makefile* detecta (comparando as datas de alteração dos arquivos fontes com as dos arquivos anteriormente compilados) qual arquivo foi alterado e compila apenas o arquivo necessário.

- Simplicidade na descrição do arquivo *makefile*: o arquivo pode ser escrito usando qualquer editor de texto puro (p.ex. *getid* no Linux ou *notepad* no Windows). Além disso a forma de utilização e sua sintaxe de criação é bastante simples.

3 Sintaxe da criação dos arquivos

Para poder utilizar o *makefile* basta criar um arquivo nomeado por *makefile* no diretório onde se encontram os arquivos fonte para compilação e executar o programa *make* no mesmo diretório.

O *makefile* é gerenciado por regras, onde cada regra tem como sintaxe:

```
regra: depend\^{e}ncias
        comando
        comando
        comando
        ...
```

- Dependências: antes de executar os comandos de uma regra, o programa *make* se certifica de que todas as dependências foram satisfeitas. Uma dependência pode ser outra regra ou então algum arquivo necessário para execução dos comandos. Por exemplo, a regra de compilação de um executável pode ter como dependência as regras que compilam as bibliotecas necessárias e também os arquivos fonte necessários.

- Lista de comandos: após as dependências, temos a lista de comandos. Note que os comandos são identados com TABs e não com espaços. Cada comando pode ser qualquer chamada de um programa de linha de comando, por exemplo, a chamada ao compilador gcc.

- Regra: a regra pode ter qualquer nome. Por exemplo, o nome da regra que compila o arquivo (programa.c) pode ser programa.exe. É ideal usar o nome do arquivo quando existe um arquivo de saída dos comandos da regra, pois é através do nome da regra que o *makefile* sabe se precisa recompilar o arquivo.

4 Um exemplo feito passo a passo

Suponha a existência dos seguintes arquivos, utilizados como exemplo e listados abaixo: “arqsoma.c”, “arqmultiplica.c”, “funcao.h” e “arqprincipal.c”.

- arqsoma.c

```
int soma(int a, int b, int c)
{
 return (a + b + c);

}
```

- arqmultiplica.c

```
int multiplica(int a, int b, int c)
{
 return (a * b * c);
}
```

- funcao.h

```
int soma(int, int, int);

int multiplica(int, int, int);
```

- arqprincipal.c

```
#include <stdio.h>
#include "funcao.h"


int main(void)
{
 int a,b,c;
 printf("Digite a,b e c\n");
 scanf("%d %d %d",&a,&b,&c);
 printf("\nSoma = %d.", soma(a, b, c));
 printf("\nMultiplicacao = %d\n\n.", multiplica(a, b, c));

 return 0;
}
```

O *makefile* para a compilação desses arquivos respeitado as dependências,linhas de comandos e regras:

```
PET-TELE: arqprincipal.o arqsoma.o arqmultiplica.o

        gcc -o programa arqprincipal.o arqsoma.o arqmultiplica.o

arqprincipal.o: arqprincipal.c

        gcc -c arqprincipal.c

arqprincipal.o: funcao.h

arqsoma.o: arqsoma.c

        gcc -c arqsoma.c

arqmultiplica.o: arqmultiplica.c

        gcc -c arqmultiplica.c
```

5 Comandos para a execução do *make*

Neste exemplo temos como comandos possíveis:

- make
- make arqprincipal.o
- make arqsoma.o
- make arqmultiplica.o

O primeiro compila todas as regras e suas dependências, e os demais são muito úteis quando apenas alguns dos arquivos são modificados.