

Tutorial de Makefile

Por: Rodrigo Moreira Silveira

O que é

O makefile é um arquivo para configuração de compilação utilizado pelo programa Make, cuja idéia é simplificar e agilizar a compilação de programas.

Vantagens do uso

- Evita a compilação de arquivos desnecessários.

Por exemplo, se seu programa utiliza 120 bibliotecas e você altera apenas uma, o makefile descobre (comparando as datas de alteração dos arquivos fontes com as dos arquivos anteriormente compilados) qual arquivo foi alterado e compila apenas a biblioteca necessária.

- Escrever um arquivo makefile é bastante simples.

O arquivo é escrito em formato texto DOS e pode ser escrito usando qualquer editor de texto puro (como o Textpad, Editplus, Crimson ou Notepad). A forma de utilização e sintaxe de criação será abordada a seguir.

Forma de utilização

Para utilizar o makefile basta criar o arquivo com o nome makefile no diretório onde se encontram os arquivos fonte para compilação e executar o programa make no mesmo diretório.

Sintaxe de criação do arquivo

O makefile funciona de acordo com regras, a sintaxe de uma regra é:

regra: dependências 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, del, copy, mkdir ou chamada ao compilador g++.

Regra:

A regra pode ter qualquer nome. Por exemplo, o nome da regra que compila o arquivo "programa.cpp" 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.

Um exemplo simples:

```
programa.exe: programa.cpp programa.h
g++ -o programa.exe programa.cpp
```

Neste makefile temos apenas uma regra que gera o "programa.exe". O programa make, antes de executar os comandos desta regra, verifica se os arquivos "programa.cpp" e "programa.h" existem. Só então são executados os comandos, no caso a compilação.

No caso da dependência ser outra regra, a regra da dependência é avaliada antes da regra dependente. Por exemplo:

```
executa: programa.exe
programa.exe

programa.exe: programa.cpp programa.h
g++ -o programa.exe programa.cpp
```

A primeira regra a ser avaliada é sempre a primeira do arquivo, no nosso caso a regra executa. Porém, durante a avaliação da regra executa, a dependência programa.exe é encontrada. Como existe uma regra com este mesmo nome no arquivo, o make sabe que se trata de uma outra regra e não de um arquivo. Antes de executar os comandos da regra executa, o make avalia a regra programa.exe, verifica as dependências conforme explicado acima e executa os comandos da regra programa.exe.

Agora a regra programa.exe está satisfeita e o make pode executar os comandos da regra executa, que no nosso caso executa o programa compilado.

Repare que neste caso, o programa só é executado se a compilação ocorrer com sucesso. Caso haja um erro de compilação, a avaliação para e o comando da regra executa não são executados.

Outro caso interessante é quando se chama o make duas vezes consecutivas. Se na primeira chamada à compilação ocorreu sem problemas, na segunda chamada o programa não é compilado, ou seja, o make percebe que não é necessário compilar o programa e pula a regra programa.exe, indo direto para a regra executa.

Um exemplo um pouco mais complexo:

```
all: msgola programa.exe
    @echo Rodando o programa
    programa.exe

msgola:
    @echo Bem vindo ao makefile. Vou tentar executar o programa.

programa.exe: programa.o
    g++ -o programa.exe programa.o -lglui -lglut32 -lopengl32

programa.o: programa.cpp programa.h
    g++ -c programa.cpp
```

O make inicia pela primeira regra no arquivo, no nosso caso a regra all. Porém a regra all tem duas dependências – msgola e programa.exe. Então o make avalia a primeira dependência, a regra msgola. Como esta regra não tem nenhuma dependência, executa os comandos da regra e vai para a próxima dependência pendente de all, ou seja, programa.exe. Porém a regra programa.exe depende da regra programa.o e programa.o depende dos arquivos “programa.cpp” e “programa.h”. Se os arquivos estiverem no mesmo diretório, as dependências da regra programa.o foram satisfeitas e podemos executar o comando da regra programa.o. Depois de executar o comando da regra programa.o, voltamos para a regra programa.exe e a executamos. Só agora temos a dependência programa.exe da regra all satisfeita e podemos executar os comandos dela. O comando “echo” repete todos os seus parâmetros (o que você escreve depois dele) na tela, a arroba antes dele apenas indica que você não quer que o comando seja repetido na tela.

Não temos que executar primeiro sempre a primeira regra que aparece. Podemos dizer para o make executar uma regra qualquer. Para isso passamos como parâmetro na linha de comando o nome da regra que queremos executar.

Por exemplo:

```
executa: programa.exe
    programa.exe

programa.exe: programa.cpp programa.h
    g++ -o programa.exe programa.cpp

limpa:
    Del programa.exe programa.o
```

Se executarmos na linha de comando:

```
make limpa
```

O make vai executar apenas as regras limpas, que elimina os arquivos compilados.