Programação de Computadores

Aula #09

A Ferramenta Make

O que é Make? Como preparar um arquivo Makefile?
Como utilizar um arquivo Makefile?

Ciência da Computação - BCC e IBM - 2024/01
Prof. Vinícius Fülber Garcia

Conceitos Básicos

O Make é um **utilitário dos sistemas Unix** que visa organizar e estruturar a compilação de um código, gerando o seu respectivo executável de forma fácil.

Geralmente, o Make é utilizado em projetos grandes de programação, facilitando a manipulação de vários arquivos de código-fonte e estabelecendo o uso correto de *flags*, por exemplo.

A aplicação Make lê um arquivo Makefile!

O Makefile

Um arquivo Makefile inclui toda a definição de comandos para a geração de um executável a partir de arquivos de código fonte e cabeçalho.

O Makefile é um arquivo estruturado em regras!

Cada regra, por sua vez, é definida (em suma) em **alvos, dependências e receitas**.

O Makefile

```
# comentário
alvo: dependência dependência dependência ...
 receita
 receita
 ...
```

- **Alvo**: nome do objeto a ser construído (geralmente um nome de arquivo ou uma ação sobre os arquivos)
- **Dependência**: alvos (ou arquivos) dos quais este alvo depende para ser construído
- **Receita**: comandos necessários para compilar/construir/gerar o arquivo alvo

O Makefile

É importante **observar a indentação** para a criação de arquivos Makefile!

A indentação deve ser feita utilizando tabulações TAB, nunca com espaços.

As receitas (ou regras) devem estar indentadas em relação ao seu respectivo alvo e dependências.

O Makefile

escreva.h

```
#ifndef __ESCREVA__
#define __ESCREVA__
void escreva (char *msg);
#endif
```

escreva.c

```
#include <stdio.h>
void escreva (char *msg)
{
  printf ("%s", msg);
}
```

main.c

```
#include "escreva.h"
int main ()
{
 escreva ("Hello, world!\n") ;
 return (0) ;
}
```

Exemplo!

Vamos considerar a criação de um arquivo Makefile para o projeto apresentado ao lado.

O Makefile

```
# Makefile para a geração do executável do projeto "escreva"
Main: main.c escreva.c escreva.h
	gcc -Wall main.c escreva.c -o hello
```

Exemplo!

O arquivo Makefile é o mais simples possível para a compilação e geração do executável para o projeto apresentado no *slide* anterior.

O Makefile

Para facilitar a manipulação de *flags*, arquivos e outros recursos, é possível **criar variáveis e atribuir dados às mesmas** para utilizá-las em diferentes pontos de um arquivo Makefile.

Definição (exemplo):

```
FLAGS = -Wall \
        -O3
```

Uso (exemplo):

```
Main: main.c escreva.c escreva.h
    gcc $(FLAGS) main.c escreva.c -o
hello
```

O Makefile

```
# Compila com flags de depuração
debug: CFLAGS += -DDEBUG -g
debug: all
```

Posteriormente, é possível utilizar as dependências de um alvo para **modificar valores de variáveis**.

Também podemos usar dependências de um alvo para indicar a execução de um conjunto de receitas (operações) definidas em outro alvo.

O Makefile

O utilitário Make mantém um conjunto de variáveis com a definição de receitas e comandos que podem ser utilizados para definir regras implícitas:

- **CC**: compilador a ser usado (por default cc) (CC = clang)
- **CPPFLAGS**: opções para o preprocessador (CPPFLAGS = -DDEBUG -I/opt/opencv/include)
- **CFLAGS**: opções para o compilador (CFLAGS = -std=c99)
- **LDFLAGS**: opções para o ligador (LDFLAGS = -static -L/opt/opencv/lib)
- **LDLIBS**: bibliotecas a ligar ao código gerado (LDLIBS = -lpthreads -lm)

O Makefile

```
# Makefile para a geração do executável do
projeto "escreva"
Main: main.c escreva.c escreva.h
      gcc -std=c99 -Wall main.c escreva.c -o
hello

# Makefile para a geração do executável do
projeto "escreva"
Main: main.c escreva.c escreva.h
      $(CC) $(CFLAGS) -Wall main.c escreva.c -o
hello
```

Exemplo!

Acima, o modelo padrão definindo as flags de compilação manualmente; abaixo, o modelo utilizando regras implícitas do utilitário Make.

A Execução do Make

Um aspecto interessante do utilitário Make é a verificação de necessidade de recompilação dos alvos definidos.

Ou seja, a ferramenta **verifica a data de modificação dos arquivos fonte, assim como do último executável gerado**, executando a compilação e gerando um novo executável apenas se necessário.

```
make: Nada a ser feito para `Main'.
```

A Execução do Make

O utilitário Make permite tem várias configurações úteis:

- **make *alvo***: constrói um alvo definido no Makefile
- **make all**: constrói todos os alvos definidos no Makefile (*default*)
- **make clean**: remove arquivos temporários (como objetos)
- **make purge**: equivalente ao clean, mas também remove arquivos intermediários (como *scripts* de configurações)
- **make debug**: semelhante ao all, mas adiciona flags de depuração (-g)
- **make test**: executa rotinas de teste, se existirem

A Execução do Make

Por padrão, qualquer execução do utilitário Make irá **parar no momento que qualquer erro acontecer**.

Porém, é possível ignorar erros e continuar a execução dos comandos na receita adicionando **o caractere “-”** no início da linha de uma receita.

Exercício #9

Clone o seguinte repositório de código:

```
git clone https://noahloomans.com/tutorials/makefile
```

1. Execute o utilitário Make uma vez e observe o resultado
2. Execute o utilitário Make mais uma vez e observe o resultado
3. Substitua a chamada do compilador GCC pela variável CC no Makefile
4. Atualize a data do arquivo main.c (touch)
5. Execute o utilitário Make e observe o resultado

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius
vinicius@inf.ufpr.br