Programação de Computadores

Aula #06-A

A Função *Main*

O que é a função *main*? Quais são os argumentos da função *main*? Como os códigos de retorno da função *main* funcionam?

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

A Função *Main*

A função *main* nada mais é que o **ponto de entrada** de um programa escrito em linguagem C.

Ou seja, é a partir dessa função que o programa resultante de um código fonte (após compilação, montagem e ligação) será executado.

Note que, independentemente da linguagem, sempre existirá um ponto de entrada... mas nem sempre uma função *main* explícita.

A Função *Main*

É possível então dizer que não é possível executar um programa em C sem que exista **uma, e somente, uma função *main*** definida em seu respectivo código-fonte.

E se a função *main* não existir?

error: linker command failed with exit code 1

A Função *Main*

Apesar de ser natural utilizar a função *main* apenas para indicar o início da execução de um programa, ela também permite a definição de argumentos com diferentes finalidades:

- `int main (void);`
- `int main (int argc, char **argv);`
- `int main (int argc, char *argv[]);`
- `int main (int argc, char **argv, char **envp);`

Os Argumentos da *Main*

Cada um dos argumentos tem uma finalidade específica dentro do corpo lógico da função *main*:

1. ***argc***: um valor inteiro; indica a quantidade de argumentos providos pelo usuário
2. ***argv***: um ponteiro de ponteiros (matriz); provê cada um dos argumentos como *strings*
3. ***envp***: um ponteiro de ponteiros (matriz); contém as variáveis de ambiente disponíveis no sistema (finalizado por um ponteiro nulo)

Os Argumentos da *Main*

Alguns *highlights* interessantes:

- O *argc* nunca é menor que um; e o *argv* nunca tem menos que um elemento
- É possível iterar os elementos de *argv* considerando o valor de *argc*
- O *envp* sempre terá, pelo menos, uma posição (apontando nulo)
- Cada registro de variável de ambiente no *envp* é dado com o seguinte formato: nome=valor
- É possível definir argc sem argv e envp, além de *argc* e *argv* sem envp

Os Argumentos da *Main*

Apesar de ser tranquilamente possível criar um laço para iterar sobre as opções providas através do *argv* utilizando *argc* (aquelas tradicionais usando o hífen), existem funções prontas para realizar tal tarefa, sendo a mais conhecida:

getopt

Essa função pertence à biblioteca ***unistd***.

A Função *getopt*

A função *getopt* é bastante simples, conta com apenas três argumentos e retorna um valor inteiro:

```
int getopt(int argc, char *const argv[],
           const char *optstring);
```

Os primeiros dois argumentos correspondem aos próprios argumentos da função *main* (*argc* e *argv*). O último argumento representa a formatação esperada para as opções disponíveis.

A Função *getopt*

Sendo assim, podemos considerar a seguinte notação para o *opt_string*:

1. Um caractere tido como opção deve estar na string
2. Se existe um valor associado obrigatório para a opção, um sinal dois pontos (:) deve aparecer depois do mesmo; se o valor for opcional, entretanto, dois sinais de dois pontos (::) devem ser incluídos

“ab:c::d”

A Função *getopt*

E, na prática, como funciona o *getopt*?

Basicamente, o **inteiro retornado indica a opção digitada** (no caso de ser um valor ASCII, ele varia entre 0 e 255, por exemplo).

Se para uma dada opção, um valor associado é previsto, então esse **valor estará indicado por um ponteiro global chamado *optarg***.

A Função *getopt*

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char **argv) {
  char next_option;
  while ((next_option = getopt(argc, argv, "a::b:")) != -1) {
    switch (next_option) {
    case 'a':
      printf("-a foi incluído, seu valor associado é: %s\n", optarg);
      break;
    case 'b':
      printf("-b foi incluído, seu valor associado é: %s\n", optarg);
      break;
    default:
      perror("%Argumentos: -a [Valor Op.] -b [Valor Ob.]\n");
      return 1;
    }
  }
  return 0;
}
```

Exemplo!

O que aconteceria com:

- ./prog.c -a123 -b321
- ./proc.c -a
- ./proc.c -b123
- ./prog.c -a -b
- ./prog.c -a -b123 -c
- ./prog.c -c

Os Retornos da *Main*

Naturalmente, sendo uma função não *void*, a *main* também pode retornar valores, nela chamados de *status* de saída.

Esse retorno em particular é recebido pelo sistema operacional e serve para indicar o *status* de finalização de execução de um programa.

Não existe um protocolo padrão para esses valores. Porém, geralmente, **o *status* zero (return 0) indica a finalização sem erros** de um programa.

Os Retornos da *Main*

Por vezes, utilizamos o retorno da função *main* quando **um processo cria um segundo processo** para executar um programa, sendo uma das formas de verificar se a execução foi bem sucedida ou não.

Mas também podemos acessar o retorno quando nós, como usuários, executamos um dado programa.

Terminal e *Shell Script*

Os Retornos da *Main*

Mas... como?

No terminal:

1. Executar o programa normalmente
2. Exibir retorno executando: *echo $?*

No Shell Script:

```
#!/bin/sh
./meu_programa
retorno=$?
echo $retorno
```

Exercício #6-A

Escreva um programa para listar o **nome das primeiras N variáveis de ambiente** recebidas pelo programa. Considere as seguintes linhas de execução como exemplo:

`./var_amb -n5`

`./var_amb -a`

No primeiro caso será exibido o nome das primeiras cinco variáveis de ambiente providas pelo sistema (se existirem cinco, se não, exibe aquelas que existem).

No segundo caso, o programa exibe o nome de todas as variáveis de ambiente.

Obrigado!

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