A função main

Video desta aula

A função `main` é o local de início (*entry point*) da execução de um código em C. Apesar de termos usado até o momento essa função sem parâmetros, ela possui alguns parâmetros que permitem a comunicação entre o programa em C e o *shell* do sistema operacional.

Protótipos

O protótipo da função `main` depende do sistema operacional subjacente:

```
// padrão C ANSI
int main (void) ;
int main (int argc, char **argv) ;
int main (int argc, char *argv[]) ;

// sistemas UNIX-like (Linux, FreeBSD, Solaris, ...) e Windows
int main (int argc, char **argv, char **envp) ;

// sistemas Apple (MacOS, iOS)
int main (int argc, char **argv, char **envp, char **apple) ;
```

Argumentos da linha de comando

Significado dos parâmetros usuais:

- `argc`: número de argumentos na linha de comando que lançou a execução;
- `argv`: vetor de strings (`char *`) contendo os argumentos da linha de comando, finalizado por um ponteiro nulo;
- `envp`: vetor de strings (`char *`) na forma "nome=valor" contendo as variáveis de ambiente exportadas pelo *shell* que lançou a execução do programa (também finalizado por um ponteiro nulo);

O código a seguir imprime na tela os argumentos usados no lançamento do programa:

argv.c

```
#include <stdio.h>

int main (int argc, char **argv, char **envp)
{
  int i ;

  printf ("Numero de argumentos: %d\n", argc) ;
```

```
  for (i = 0; i < argc; i++)
    printf ("argv[%d]: %s\n", i, argv[i]) ;

  return (0) ;
}
```

Um exemplo de compilação e execução do código acima:

```
$ gcc argv.c -o argv -Wall

$ ./argv teste 1 2 3 --help
Numero de argumentos: 6
argv[0]: ./argv
argv[1]: teste
argv[2]: 1
argv[3]: 2
argv[4]: 3
argv[5]: --help
```

Funções específicas

Para ler e tratar mais facilmente as opções da linha de comando informadas por *argc/argv*, sugere-se usar funções já prontas para isso, como `getopt` ou `arg_parse` (link)

Eis um exemplo de uso da função `getopt` para a leitura de opções da linha de comando, adaptado do manual da GNU-LibC:

options.c

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char **argv)
{
  int flag_a = 0;
  int flag_b = 0;
  char *value_c = NULL;
  int option;

  opterr = 0;

  // options:  -a, -b, -c value (defined by "abc:")
  while ((option = getopt (argc, argv, "abc:")) != -1)
    switch (option)
      {
      case 'a':      // option -a was set
        flag_a = 1;
        break;
      case 'b':      // option -b was set
        flag_b = 1;
        break;
```

```
      case 'c':      // option -c was set with value
        value_c = optarg;
        break;
     default:
        fprintf (stderr, "Usage: %s -a -b -c value\n", argv[0]);
    exit (1) ;
      }

  printf ("flag_a = %d, flag_b = %d, value_c = %s\n",
          flag_a, flag_b, value_c);

  return 0;
}
```

Status de saída

Outro canal de interação importante entre o programa C e o sistema operacional é o valor de retorno da função `main`, que é devolvido ao SO após a execução na forma de um *status* de saída (*exit status*).

retval.c

```
#include <stdio.h>

int main (int argc, char **argv, char **envp)
{
  return (14) ;
}
```

O *status* de saída de um processo pode ser consultado no terminal UNIX (shell Bash) através da variavel `$?` disponível no shell:

```
$ gcc retval.c -o retval -Wall
$ ./retval
$ echo $?
14
```

O status de saída também pode ser usado em scripts do shell:

testa-retorno.sh

```
#!/bin/sh

if retval
then
  # exit status is zero
  echo "true"
else
  # exit status is NOT zero
  echo "false"
fi
```

Exercícios

1. Escrever um programa que recebe uma lista de parâmetros na linha de comando. Ele não escreve nada na saída, mas devolve como *status* de saída o número de parâmetros que iniciam com o caractere "`-`".
2. Escrever um programa `exists`, que recebe um nome (ou caminho) de arquivo na linha de comando e devolve, no *status* de saída, 0 se o arquivo existe ou 1 se ele não existe.
3. Escrever um programa para listar as variáveis de ambiente recebidas pelo programa (parâmetro `envp` da função `main`); essas variáveis podem ser consultadas no terminal (*shell*) através do comando `env`.