2 Um Primeiro Programa em C

Ronaldo F. Hashimoto, Carlos H. Morimoto e José A. R. Soares

O objetivo dessa aula é introduzir você à linguagem C em ambiente Linux, primeiramente mostrando a sua estrutura, e a seguir com um exemplo simples. Antes de iniciar essa aula, é desejável que você disponha de um editor de texto para escrever o programa, e verifique também a existência do compilador gcc em seu sistema.

Ao final dessa aula você deve ser capaz de:

- Descrever o ambiente em que os seus programas serão desenvolvidos.
- Escrever um programa em C que recebe dados do teclado e imprime dados na tela, usando as funções `scanf` e `printf`.
- Compilar um programa em C usando o gcc.
- Executar o programa após a compilação.

2.1 Sobre Linux

Todo programa roda em um ambiente definido pelo conjunto de hardware e software a sua disposição. Como mais de 90% dos computadores pessoais (PCs) no mundo usam algum sistema operacional da Microsoft, você provavelmente não sabe o que é Linux, e muito menos ainda sabe por que esses professores de Computação teimam em insistir que há alguma vantagem em usar Linux em seu PC, quando você está muito satisfeito (ou confortável) com o sistema que você possui agora.

O Linux começou a ser desenvolvido em 1991 por Linus Torvalds e é baseado no sistema operacional Unix. Esse projeto pode ser considerado hoje como um dos melhores exemplos de sucesso no desenvolvimento de software aberto (e que é grátis!). A maior aplicação do Linux se encontra em servidores (essas máquinas que mantem a Internet no ar) e por isso muitas companhias já apoiam esse sistema, como a Dell, HP e a IBM, entre outras. Como exemplo de usuário podemos citar a Google, que (estima-se) possui cerca de 450.000 servidores rodando Linux. Além de servidores, o Linux é utilizado também em supercomputadores, em plataformas de jogos como o PlayStation 2 e 3, em telefones celulares, e muitos outros sistemas computacionais. No entanto, apenas cerca de 1% dos desktops rodam Linux.

Para aprender a programar em C você não precisa instalar Linux em seu computador, pois há várias alternativas de ferramentas que você pode usar para desenvolver seus programas no ambiente Windows que seu professor pode lhe indicar. Porém, essa é uma excelente oportunidade de conhecer o Linux, o que lhe pode trazer uma grande vantagem profissional no futuro (assim como, por exemplo, talvez seja importante aprender inglês e chinês). Um exemplo de projeto de grande porte que pode impulsionar ainda mais o uso de Linux é o projeto OLPC (one laptop per child), também conhecido como laptop de 100 dólares, que tem o potencial de atingir milhões de crianças em todo o mundo.

2.2 Interfaces Gráficas x Linha de Comando

Você provavelmente já está familiarizado com interfaces gráficas, essas que apresentam janelas, menus, ícones e outros componentes gráficos que você pode clicar, um cursor que você controla com o mouse, etc. Essas interfaces foram desenvolvidas na década de 1990, sendo que na década de 1980 as melhores interfaces eram do tipo `linha de comando`. Nesse tipo de interface, o monitor, em geral verde, era capaz de apresentar apenas texto. Dessa forma o usuário precisava digitar o nome do comando a ser executado pelo computador.

Atualmente, no Linux, você tem aplicações do tipo `Terminal` que criam uma janela no ambiente gráfico onde você pode entrar com comandos usando o teclado [2]. É no `Terminal`, que você vai compilar e executar o seu

[2] Veja em nossa página como fazer isso no Windows, usando por exemplo uma janela do CYGWIN

programa.

2.3 Sistema de Arquivos

As informações que você possui no computador são armazenadas nos dispositivos de memória não volátil na forma de arquivos. De forma geral, podemos definir 3 tipos de arquivos: diretórios, dados e aplicativos. Diretórios são arquivos que contém outros arquivos e permitem que você organize todas as informações em seu disco (HD). Os outros tipos de arquivos contem informações. A diferença básica entre eles é que os aplicativos podem ser executados pelo computador, enquanto os dados (as vezes chamados de documentos) são utilizados como entrada e/ou saída dos aplicativos.

O compilador gcc, por exemplo, é um aplicativo [3] que recebe como entrada um arquivo fonte, e gera um arquivo executável (um outro aplicativo). O editor de texto é um exemplo de outro tipo de aplicativo, que não necessariamente precisa receber um arquivo de entrada, e pode gerar arquivos de dados na saída. Digamos que você use um editor de texto para escrever um programa em C, e salva esse programa no arquivo "exemplo.c". Embora esse arquivo contenha um programa, ele não pode ser executado enquanto não for traduzido para linguagem de máquina pelo compilador gcc.

Para esse curso, recomendamos que você sempre rode o gcc com as seguintes opções: "-Wall -ansi -O2 -pedantic". Essas opções garantem que o gcc vai lhe fornecer todas os avisos que ele é capaz de gerar para prevenir você contra possíveis falhas no seu programa. Assim, para compilar o arquivo "exemplo.c", podemos utilizar o seguinte comando na janela Terminal:

```
gcc -Wall -ansi -O2 -pedantic exemplo.c -o exemplo
```

A opção "-o" indica o nome do arquivo de saída, no caso, apenas "exemplo", sem nenhuma extensão.

2.4 O Esqueleto de um Programa em C

Finalmente, vamos ver qual o conteúdo de um arquivo com um programa em C. Para que você consiga compilar o seu programa em C sem problemas utilizando o gcc, todos os seus programas devem possuir o seguinte esqueleto:

```
1 #include <stdio.h>
2
3 int main()
4 {
5   /* declaração de variáveis */
6
7   /* lista de comandos */
8
9   return 0;
10 }
```

Por enquanto considere esse esqueleto como uma "receita de bolo", ou seja, todo programa em C deve conter os comandos das linhas $1, 3, 4, 9$ e 10.

2.5 Exemplo de um Programa

Para entendermos melhor, considere o programa em C apresentado na Fig. 2. Esse programa faz uma pergunta ao usuário (quantos anos você tem?), espera que o usuário entre com uma resposta numérica através do teclado,

[3] Muitas vezes chamamos aplicativos de programas, mas isso seria confuso em nosso contexto já que os programas que você vai escrever em C não podem ser executados antes de compilados.

e finaliza com um comentário sobre a idade que depende da resposta. Lembre que em C, assim como nos microprocessadores, as instruções são executadas sequencialmente, uma de cada vez.

```
1  # include <stdio.h>
2
3  int main() {
4
5    /* Primeiro programa em C */
6
7    /* declarações: todas as variáveis utilizadas precisam ser declaradas */
8
9    int idade;
10
11   /* início do programa */
12
13   printf ("Quantos anos voce tem?: ");
14   scanf ("%d", &idade);
15
16   printf ("%d? Puxa, voce parece que tem so %d anos!\n", idade, idade * 2);
17
18   /* fim do programa */
19
20   return 0;
21 }
```

Figura 2: Primeiro Programa

2.5.1 Comentários

Primeiramente, os textos entre os símbolos `/*` e `*/` (linhas $5, 7, 11$ e 18) são *comentários*. Comentários não interferem no programa, mas auxiliam os programadores a entender e documentar o código.

2.5.2 Declaração de Variáveis

Todo programa precisa de espaço na memória para poder trabalhar e as declarações reservam o espaço necessário para isso. Na linha 9, temos uma declaração de uma variável de nome `idade`. Esta variável guarda números de tipo **int** (inteiro). Em C, todas as variáveis utilizadas precisam ser declaradas no início de cada bloco de comandos. A forma de declaração de variáveis é:

int <nome_da_variavel>;

2.5.3 Funções de Leitura e Impressão

Todos os seus programas devem se comunicar com o usuário através de funções de impressão (na tela) e de leitura (pelo teclado). Basicamente, nos nossos programas, o usuário fornece números inteiros para o programa através da leitura pelo teclado (função `scanf`); enquanto que o programa fornece ao usuário os resultados via impressão de mensagens na tela (função `printf`). No nosso exemplo, a função de impressão na tela está sendo utilizada nas linhas 13 e 16; enquanto que a função de leitura pelo teclado está sendo utilizada na linha 14.

2.5.4 Função de Impressão na Tela

Basicamente, a função `printf` imprime todos os caracteres que estão entre aspas. Assim, o `printf` da linha 13 imprime a mensagem (sem as aspas) `"Quantos anos voce tem?: "`. Note o espaço em branco no final da mensagem que também é impresso!

Agora observe o `printf` da linha 16. Este `printf` tem duas diferenças com relação ao `printf` da linha 13. A primeira diferença é que dentro da mensagem do `printf` da linha 16 (caracteres que estão entre aspas) podemos encontrar duas sequências de caracteres: `"%d"` e `"\n"`. Além disso, depois da mensagem, temos duas expressões aritméticas envolvendo a variável `idade` separadas por vírgulas: (a) `"idade"` (seria como a expressão aritmética `"idade * 1"`) ; and (b) a expressão aritmética `"idade * 2"`.

O `printf` da linha 16 imprime na tela todos os caracteres que estão entre aspas, com exceção da sequência de caracteres `"%d"` e `"\n"`.

Para cada sequência de caracteres `"%d"`, a função `printf` imprime na tela um número inteiro que é resultado das expressões aritméticas contidas no `printf` separadas por vírgula. Assim, o primeiro `"%d"` imprime na tela o conteúdo da variável `"idade"` e segundo `"%d"` imprime na tela o resultado da expressão `"idade * 2"` (uma vez que a expressão `"idade"` vem antes da expressão `"idade * 2"` no `printf` da linha 16.

A sequência de caracteres `"\n"`, indica à função `printf` para "pular de linha", isto é, faz com que o cursor da tela vá para a próxima linha. No `printf` da linha 16, como a sequência está no final da mensagem, isto significa que depois de imprimir a mesma na tela, o cursor irá para a próxima linha.

2.5.5 Função de Leitura pelo Teclado

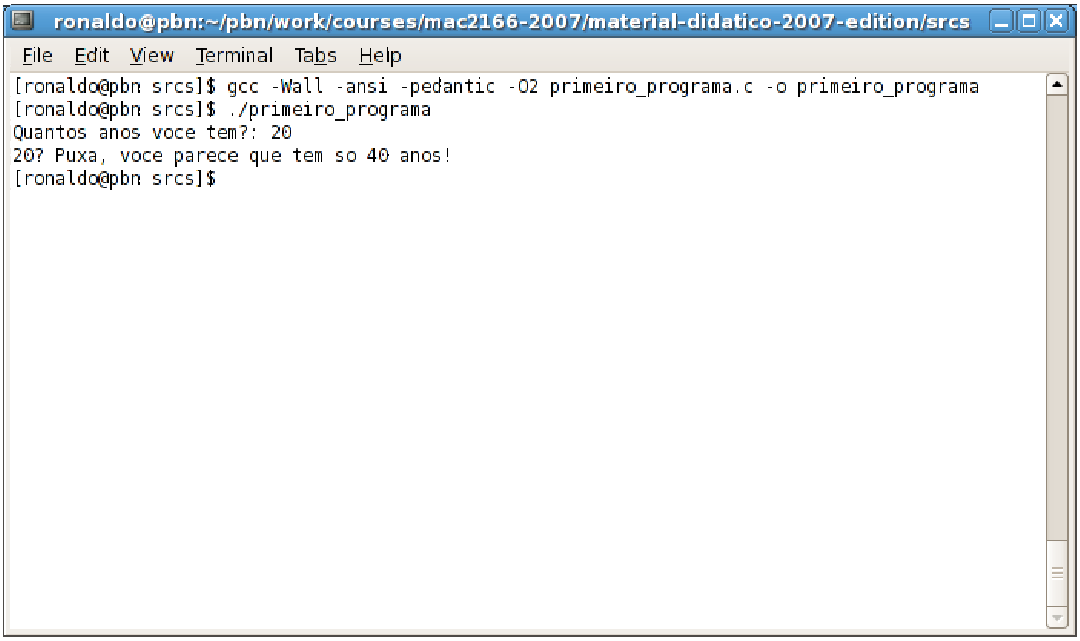
Para ler um número inteiro pelo teclado, você deve usar a função `scanf` da seguninte forma:

```
scanf ("%d", &<nome_da_variavel>);
```

o `scanf` irá esperar o usuário digitar um número inteiro pelo teclado e, após o usuário digitar a tecla `<ENTER>`, armazenará o número digitado na variável `<nome_da_variavel>`. Um exemplo está na linha 14 do primeiro programa: o número digitado irá ser armazenado na variável `idade`. Observe que no `scanf` deve-se colocar o caractere `"&"` antes do nome da variável.

2.5.6 Retornando ao nosso Exemplo

Executando o programa, temos:



Observe que:

1. O número `"20"` que aparece depois da mensagem

   ```
   "Quantos anos voce tem?:  "
   ```

   foi digitado pelo usuário e lido pela função `scanf`.

2. Este número `"20"` aparece ao lado da mensagem, pois o `printf` que a imprime na tela não tem a sequência de caracteres `\n` no final; caso contrário, o número `"20"` seria digitado na próxima linha.

3. Uma vez que o usuário, depois de digitar o número 20, deve dar um `<ENTER>`, o cursor automaticamente irá para a próxima linha; observe que a mensagem

   ```
   "20?  Puxa voce parece que tem so 40 anos!"
   ```

   aparece na próxima linha.

4. Os números `"20"` e `"40"` (resultados das expressões aritméticas `"idade"` e `"idade * 2"`) são colocados no lugar do `%d` do segundo `printf` do programa.

2.5.7 Impressão de %d e \n

Para imprimir na tela a sequência de caracteres `"%d"`, você deve usar

```
printf ("%%d");
```

e para imprimir `"\n"`, você de usar

```
printf ("\\n");
```

2.6 Mais detalhes sobre o esqueleto

A linguagem C é uma linguagem de alto nível criada por Brian Kernighan e Dennis Ritchie no início da década de 1970 nos laboratórios da AT&T Bell, e suas origens estão relacionadas ao desenvolvimento do sistema operacional Unix. O C é também uma linguagem estruturada, que permite que um problema complexo seja facilmente decomposto em problemas mais simples, definindo assim os módulos usando termos próximos à linguagem natural (embora em inglês).

Cada módulo básico é chamado de função, e cada função precisa ter um nome (ou identificador) bem definido e diferente dos demais. No caso, a função de nome `main`é necessária em todos os programas pois define o início da execução do programa. A função `main`foi definida no esqueleto como uma função `int` (ou seja, inteira), e por isso precisa devolver um valor inteiro. Daí a necessidade do comando `return 0`, apenas por consistência, já que o zero não é realmente utilizado. Toda função em C recebe também parâmetros. Por exemplo uma função seno deve receber como parâmetro um ângulo. A lista de parâmetros em C é declarada entre parênteses depois do nome e, no caso da função `main`, ela recebe zero parâmetros pois não há nada entre os parênteses. As chaves definem o início e fim de um bloco de instruções.

Embora os comandos da linguagem C sejam bem poderosos, eles são limitados. Mas com a maturidade de uma linguagem, vários programadores desenvolveram funções auxiliares que facilitam a programação e podem ser compartilhados com outros programadores. Para utilizar essas funções, basta que você especifique onde encontrá-las através das linhas de `include`. No caso, o pacote `stdio.h` contem as rotinas necessárias para ler caracteres do teclado e imprimir caracteres no monitor, ou seja, contem as funções `scanf` e `printf`.