Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Laboratório de Sistemas Operacionais

Laboratório 03 – Chamadas de Sistema

Programas em Linguagem C no Linux

O sistema operacional Linux foi desenvolvido em linguagem C e a maioria de seus aplicativos são desenvolvidos nesta linguagem.

Para executar alguns comandos que acessam o kernel do sistema operacional Linux é necessário que programas em linguagem C sejam desenvolvidos utilizando algum compilador C para Linux. O compilador GCC é o mais popular e amplamente utilizado para a criação de programas executáveis no Linux utilizando a linguagem C.

Caso ainda não tenha instalado o GCC em sua distribuição Linux, siga os passos abaixo para instalar o gcc, g++ e o make, dentre outros programas que são utilizados para construção de programas.

1) **sudo apt update**
2) **sudo apt install build-essential**
    a. Responda Y ou S para a pergunta.
3) **gcc --version**

```
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

A sintaxe de utilização do GCC é:

gcc arquivo_fonte.c -o arquivo_executável.

O nome do arquivo executável é opcional. Caso não seja apresentado o nome do arquivo executável, por padrão, um arquivo com o nome **a.out** será criado, ou seja, **a.out** é o arquivo executável que o GCC tem como saída padrão.

A execução do arquivo executável gerado pelo GCC deve ser efetuada da seguinte forma:

./arquivo_executável
./a.out

Como exemplo, vamos criar um programa em linguagem C, compilar e executar.

```
#include <stdio.h>
/* Isto é um comentário */
int main() {
printf("Hello, world!\n");
return 0;
}
```

1) Editar usando um editor de texto do Linux (GEdit, nano, vi, etc.) e salvar com a extensão "**.c**"
2) Compilar usando o seguinte comando
    a. **gcc arquivo_fonte.c -o primeiro**
3) Executar o arquivo criando anteriormente
    a. **./primeiro**

Chamadas de sistema

As chamadas de sistema são formas de acessar funções existentes no kernel de forma segura. Através de um conjunto de funções é possível acessar funcionalidades que o sistema operacional possui sem interferir na execução de outros processos.

As chamadas de sistema (*system calls*) no sistema operacional Linux são descritas no seguinte link: http://man7.org/linux/man-pages/man2/syscalls.2.html.

Para descrever como as chamadas de sistema são usadas na linguagem C, veremos alguns exemplos sobre como acessar arquivos e comandos.

1) Editar o arquivo abaixo em um editor de texto no Linux

```
#include <stdio.h>
#include <fcntl.h>

int main(){
    char buff[100], fn[10];
    int fd, n;

    printf("Entre com o nome de um arquivo\n");
    scanf("%s", fn);
    fd = open(fn, O_RDONLY);
    n = read(fd, buff,100);
    n = write(1, buff,n);
    printf("\n\nCaracteres lidos e exibidos%d\n",n);
    close(fd);
}
```

2) Compilar e executar o arquivo
   a. As funções **open**, **read**, **write** e **close** utilizam chamadas de sistema para acessar o sistema de gerenciamento de arquivos do Linux
      i. **open**: abre um arquivo;
      ii. **read**: lê os dados de um arquivo;
      iii. **write**: escreve dados em um arquivo;
      iv. **close**: fecha o arquivo.
   b. O programa precisa de um arquivo que possua dados para serem lidos. Esse arquivo é informado através da variável **fn**.
   c. A chamada de sistema **open** cria um ponteiro **fd** para o arquivo indicado em **fn**. Neste caso a opção **O_RDONLY** indica que o arquivo será somente para leitura.
   d. A chamada de sistema **read** acessa 100 bytes do arquivo apontado por **fd** e coloca esses dados na variável **buff**, onde **n** é o número de bytes lidos do arquivo apontado por **fd**.
   e. A chamada de sistema **write** escreve o conteúdo da variável **buff** na saída padrão, indicado por **1**, até o limite apontado por **n** (número de bytes).
   f. A chamada de sistema **close** efetua o fechamento correto do arquivo. Caso existam dados no buffer de saída para serem escritos no arquivo, estes dados serão escritos antes do arquivo ser fechado.
   g. ./arquivo_executável
      i. Informar um nome de arquivo que contenha algum dado.
         1. Caso necessário, criar este arquivo antes da execução.

TAREFA

1) Executar e explicar as ações que os programas abaixo executam
    a. Programa 01

```
#include <stdio.h>
#include <fcntl.h>
#include <string.h>

int main(){
   char *buff, fn[10];
   int fd, n, i;

   printf("Entre com o nome de um arquivo novo:\n");
   scanf("%s",fn);
   printf("\nEntre com um texto qualquer:");
   scanf("%s",buff);
   fgets(buff,100,stdin);
   fd = open(fn,O_CREAT|O_WRONLY);
   n = write(fd,buff,strlen(buff));
   close(fd);

   printf("\n\nVeja o conteúdo do arquivo criado\n");
}
```

    b. Programa 02

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>

int main(){
      char d[10];
      DIR *p;
      struct dirent *d1;

      printf("Entre com um nome de um diretório:\n");
      scanf("%s",d);
      p = opendir(d);
      if( p==NULL ){
          perror("Diretório não encontrado");
          exit(-1);
      }
      while( d1 = readdir(p) )
          printf("%s\n",d1->d_name);
}
```

2) Consulte o material destacado no link abaixo e identifique as chamadas de sistema que trabalhem com diretórios e quais executam busca em arquivos.
    a. http://man7.org/linux/man-pages/man2/syscalls.2.html