Lista de Exercícios Práticos  
**Manipulando Arquivos com Chamadas de Sistema**

Laboratório de Sistemas Operacionais

**INTRODUÇÃO**

Nesta lista de exercícios, você utilizará algumas syscalls básicas em C para manipular arquivos. **Submeta um arquivo .c no Moodle para cada exercício.**

Segue uma apresentação e revisão de alguns conceitos necessários, incluindo os parâmetros e tipos utilizados nas chamadas de sistema open, read, write e close.

**Syscalls**

**open( )**

* **Função:** Abre um arquivo e retorna um descritor de arquivo (file descriptor) do tipo int.
* **Protótipo:**



**int fd = open(**



**const char \*<caminho do arquivo>,**

**int <modo de abertura>,**

**mode\_t <permissões do arq.> (opcional)**

**);**

* **Obs.:** Os modos de abertura podem ser:

**O\_RDONLY -** Abre apenas para leitura. **O\_WRONLY -** Abre apenas para escrita. **O\_RDWR -** Abre para leitura e escrita.

**O\_CREAT -** Cria o arquivo caso não exista. **Entre outros (consulte a documentação).**

**read( )**

* **Função:** Lê dados de um arquivo para um buffer. Retorna um valor do tipo ssize\_t, contendo o tamanho (em bytes) da última leitura realizada.
* **Protótipo:**



**ssize\_t bytesLidos = read(**



**int <descritor de arquivo>,**

**void \*<buffer que receberá os bytes>,**

**size\_t <tamanho maximo a ser lido>**

**);**

* **Obs.:** Ao lidar com a leitura de arquivos que serão processados como strings, para o terceiro parâmetro é recomendado que se use o padrão sizeof(buffer) – 1 para garantir que haverá espaço ao final do buffer para inserir o caracter nulo \0, essencial para que funções que processam strings (como printf) saibam onde uma string acaba.

**write( )**

* **Função:** Escreve dados de um buffer para um arquivo. Retorna um valor do tipo ssize\_t, que indica o número de bytes efetivamente escritos.
* **Protótipo:**



**ssize\_t bytesEscritos = write(**

**int <descritor de arquivo>,**

**const void \*<buffer com os dados a serem escritos>, size\_t <tamanho dos dados a serem escritos>**

**);**

* **Obs.:** A syscall write() escreve o conteúdo do buffer no arquivo a partir da posição atual do ponteiro de escrita. Se o arquivo foi aberto com a flag O\_APPEND, a escrita será realizada no final do arquivo. Caso contrário, a escrita ocorrerá na posição indicada pelo ponteiro de escrita, que avança conforme os dados são gravados.

**close( )**

* **Função: Fecha um arquivo previamente aberto.**
* **Protótipo:**



**int status = close(**



**int <descritor de arquivo>**

**Buffer**

Trata-se de uma área de memória temporária que utilizamos para guardar dados durante a transferência entre diferentes partes de um programa, ou entre o programa e o sistema operacional.

**Descritor de Arquivo**

Quando um arquivo é aberto, o SO retorna um descritor de arquivo (fd). A cada leitura realizada em um dado arquivo, um ponteiro de leitura/escrita é utilizado para rastrear a posição atual no arquivo a partir da qual a próxima leitura/escrita será realizada.

**Bibliotecas**

Para realizar os exercícios, as seguintes bibliotecas precisarão ser importadas:



**#include <fcntl.h> // Inclui as definições para open e suas flags**

**#include <unistd.h> // Inclui as definições para read, write e close**

**#include <stdio.h> // Inclui as definições para printf**



**EXERCÍCIOS**

**Exercício 1: Abrir e Ler um Arquivo**

Escreva um programa em C que realize as seguintes operações:

1. Abra um arquivo de texto chamado ‘lab2\_ex1.txt’ em modo leitura.
2. Verifique se o arquivo foi aberto com sucesso. Se não, exiba uma mensagem de erro e encerre o programa.
3. Leia o conteúdo do arquivo para um buffer.
4. Exiba o conteúdo lido na tela.
5. Feche o arquivo após a leitura.

**Requisitos:**

* O buffer deve ter um tamanho de 128 bytes.
* Certifique-se de adicionar um caractere nulo (\0) ao final do buffer após a leitura.

**Exercício 2: Contar Linhas em um Arquivo**

Escreva um programa em C que:

1. Abra um arquivo de texto (lab2\_ex2.txt) e leia seu conteúdo.
2. Conte quantas linhas existem no arquivo.
3. Exiba o número total de linhas na tela.

**Requisitos:**

* Utilize um buffer de 128 bytes para ler o arquivo em partes.
* A contagem de linhas deve ser feita identificando os caracteres de quebra de linha (\n).
* O loop deve continuar até que todo o arquivo tenha sido processado.

**Exercício 3: Copiar o Conteúdo de um Arquivo para Outro**

Escreva um programa em C que:

1. Abra um arquivo de origem (lab2\_ex3\_origem.txt) em modo leitura (O\_RDONLY).
2. Abra ou crie um arquivo de destino (lab2\_ex3\_destino.txt) em modo escrita (o parametro do modo de abertura deve ser ‘O\_WRONLY | O\_CREAT | O\_TRUNC’).
3. Leia o conteúdo do arquivo de origem usando read() e escreva no arquivo de destino usando write().
4. Feche ambos os arquivos após a operação.

**Requisitos:**

* Use um buffer de 128 bytes para copiar o conteúdo.
* Verifique se ambos os arquivos foram abertos com sucesso e trate possíveis erros.
* Utilize um loop para ler o conteúdo do arquivo de origem e escrevê-lo no arquivo de destino até que todo o conteúdo tenha sido copiado.