

# Sistemas Operacionais II

## *Aula 1*

Autor: Renê de Souza Pinto

Orientação: Prof. Dr. Francisco José Monaco

`rene@grad.icmc.usp.br`, `monaco@icmc.usp.br`

Universidade de São Paulo

Instituto de Ciências Matemáticas e Computação - ICMC

Escola de Engenharia de São Carlos - EESC

# Sumario

- Revisão teórica
- Sistemas de Arquivos
- Prática 1: Chamadas de sistemas

# Revisao

- O que é um sistema operacional (S.O.)?

# Revisao

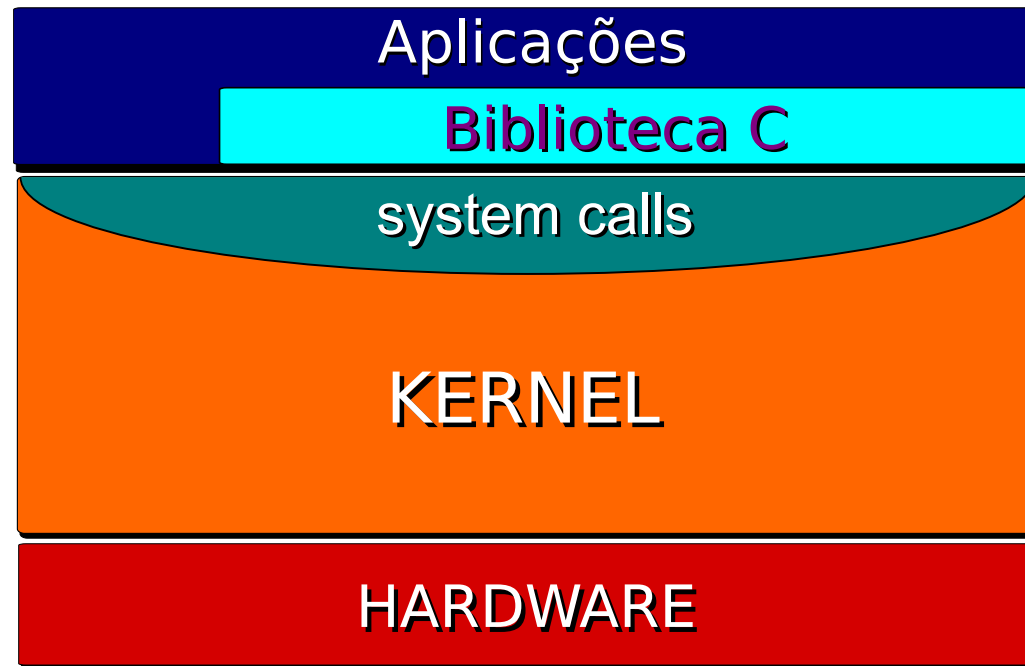
- O que é um sistema operacional (S.O.)?
- Segundo Tanenbaum[1]:
  - Gerenciador de Recursos:  
Processador, Memória, Discos, Teclado, Vídeo,  
todos os periféricos da máquina devem ser  
gerenciados

# Revisao

- O que é um sistema operacional (S.O.)?
- Segundo Tanenbaum[1]:
  - Gerenciador de Recursos:  
Processador, Memória, Discos, Teclado, Vídeo, todos os periféricos da máquina devem ser gerenciados
  - Máquina Virtual (ou Máquina Extendida):  
Abstrai o hardware para o programador provendo uma estrutura padronizada: Sistema de arquivos e chamadas de sistemas (*syscalls*)

# Visao em Camadas

- S.O.: Camada entre o hardware e os aplicativos



# Camadas

- Hardware: É a parte física da máquina: Processador, memória, controladoras, etc.

# Camadas

- Kernel: É o núcleo do sistema, gerencia todo o hardware assim como provê as chamadas de sistemas (*syscalls*). Pode ser visto como o S.O. em si.
  - Micro-kernel: Baseado no paradigma cliente-servidor. Várias funcionalidades são providas por softwares externos ao kernel, que se comunicam com o mesmo.
  - Monolítico: Todas as funcionalidades são implementadas e executadas dentro do kernel.

# Camadas

- Aplicações: É a camada que contém os aplicativos de usuário. A biblioteca C provê uma interface padronizada, permitindo a portabilidade entre sistemas, ou seja, possibilitando que softwares sejam compilados e executados em diversas plataformas (ex: Linux, FreeBSD, Solaris) sem modificação de código.

# Exemplos de S.O.

- Sistemas Operacionais bem distintos:
  - MS-DOS - **M**icrosoft **D**isk **O**perating **S**ystem [3]
  - Microsoft Windows® [5]
  - Linux [6]

# MS-DOS

- **MS-DOS - Microsoft Disk Operating System**
  - Funciona no Modo Real da arquitetura x86 fornecendo funções para os programadores através de um conjunto de interrupções. Acesso a memória estendida ou funcionalidades mais avançadas devem ser implementados através de softwares específicos.
  - Acesso somente a 1MB de memória (limitação imposta pelo modo Real do x86)
  - Acesso a memória estendida (> 1MB) com softwares específicos (ex: HIMEM.SYS).

# Ola Mundo no MS-DOS

```
;
; Ola Mundo para o MS-DOS
;
; Autor: Renê de Souza Pinto
; Data.: Fev, 2009
;
; Para compilar: nasm hello.asm -f bin -o hello.com
;
```

[Bits 16]

```
; Programas .COM começam em 100h de offset, pois
; 0-100h contém os parâmetros passados por linha
; de comando
ORG 0x100
```

# Ola Mundo no MS-DOS

```
section .text
```

```
; Carrega endereço de offset da string.  
; Não é necessário carregar o endereço de  
; segmento porque quando o arquivo .COM é  
; carregado, todos os registradores de segmento  
; contêm o mesmo valor  
lea ax,[message]  
mov dx,ax
```

# Ola Mundo no MS-DOS

```
; Imprime mensagem
xor ax,ax      ; limpa AX
mov ah,0x09    ; 02h - Escreve string na saída padrão
int 0x21       ; Chama interrupção do DOS

; Sai do programa
mov al,0x00    ; Código de saída
mov ah,0x4C    ; 4Ch - Sair do programa
int 0x21       ; Chama interrupção do DOS

section .data

message db 'Ola Mundo!$'
```

# Windows

- Microsoft Windows®
  - Conjunto de sistemas operacionais da Microsoft
  - Dominante no mercado de desktops
  - Existe em várias versões: 3.11, 95, 98, XP, 2000, 2003 Server, NT, Vista
- Microsoft Windows NT®
  - Possui um Kernel híbrido

- Linux
  - Criado pelo finlandês Linus Torvalds
  - Licenciado pela GPL v.2 (código-fonte aberto)
  - Possui milhares de desenvolvedores espalhados ao redor do mundo
  - Kernel monolítico. Primeira versão foi baseada no Minix, um S.O. *Unix Like* desenvolvido por Andrew S. Tanenbaum

# Ola Mundo no Linux

```
;
; Ola Mundo para o Linux
;
; Autor: Renê de Souza Pinto
; Data.: Fev, 2009
;
; Para compilar: nasm hello.asm -f elf
; Para linkar:   ld hello.o -o hello
;

[Bits 32]

[global _start]

%define SYS_WRITE 0x04 ; índice para a syscall sys_write
%define SYS_EXIT  0x01 ; índice para a syscall sys_exit
%define STDOUT   0x01 ; descritor da saída padrão
```

# Ola Mundo no Linux

```
section .text
```

```
_start:
```

```
    ; Imprime mensagem
```

```
    ; syscall sys_write arguments
```

```
    mov edx, msglen
```

```
    mov ecx, message
```

```
    mov ebx, STDOUT
```

```
    ; Chama interrupção de syscall do Linux
```

```
    mov eax, SYS_WRITE
```

```
    int 0x80
```

# Ola Mundo no Linux

```
    ; Sai do programa
    ; Empilha argumentos da syscall sys_exit
    mov ebx,0x00          ; Código de retorno
    push ebx

    ; Chama interrupção de syscall do Linux
    mov eax,SYS_EXIT
    int 0x80
```

```
section .data
```

```
message db 'Ola Mundo!',13,10,0
msglen  equ $-message
```

# Ambiente Unix

- Nosso estudo será feito no Ambiente Unix (padrão POSIX):
  - POSIX - Portable Operating System Interface (o X é de Unix): Padrão consolidado
  - Vários sistemas operacionais são baseados ou implementam este padrão: Linux, FreeBSD, OpenSolaris, AIX, HP-UX, IRIX, Minix, QNX

# Ambiente Unix

- Nosso estudo será feito no Ambiente Unix (padrão POSIX):
  - Sistemas Operacionais com código-fonte aberto: Permite explorarmos a fundo os conceitos teóricos e analisarmos implementações reais

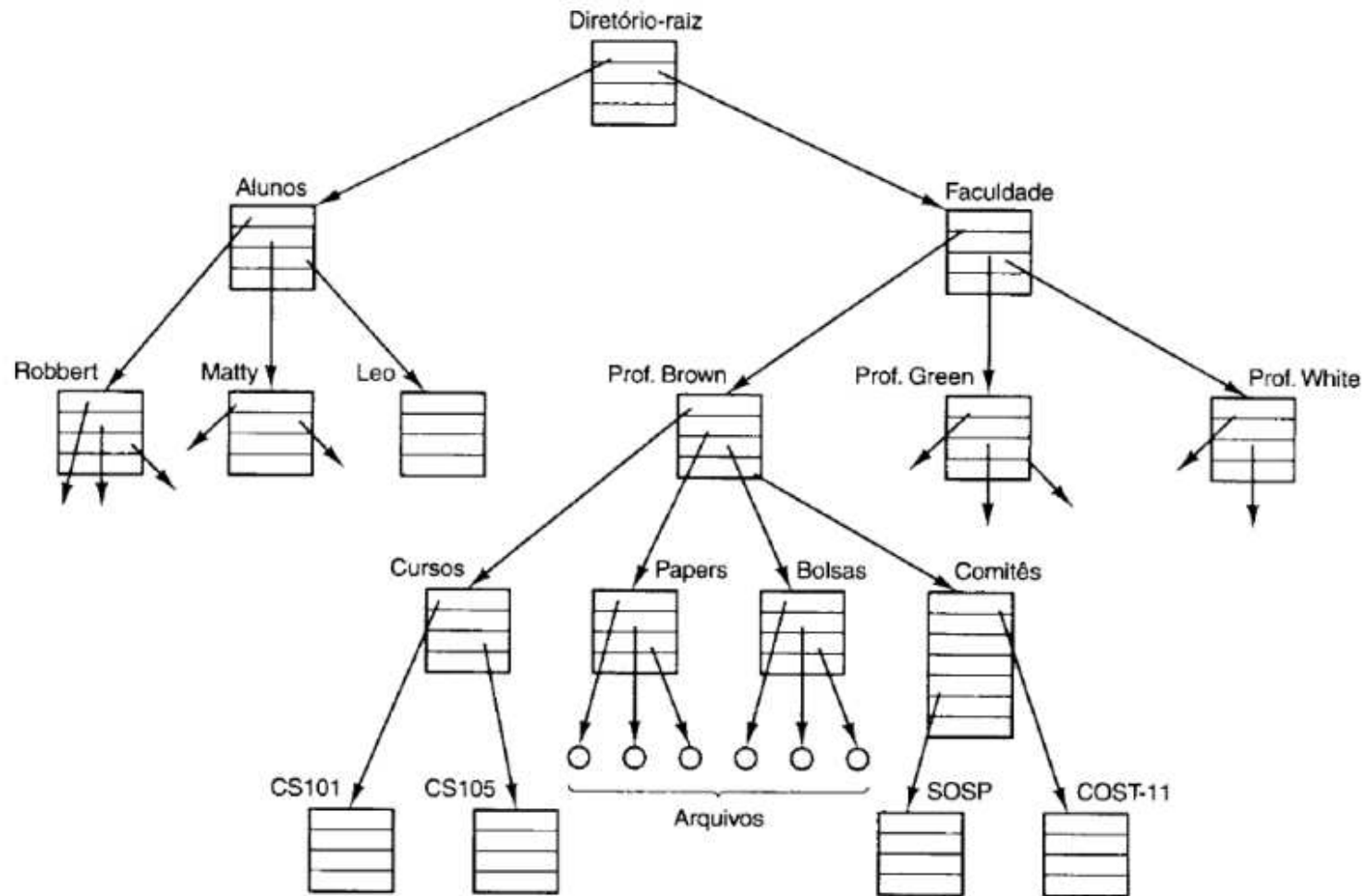


# Ambiente Unix

# Sistemas de Arquivos

- Sistema de arquivos: É a maneira com a qual os dados são organizados nos dispositivos de armazenamento (discos, SSD, etc) e abstraídos pelo sistema.
- No Unix tudo é um arquivo!
- Arquivos e diretórios são organizados em árvore

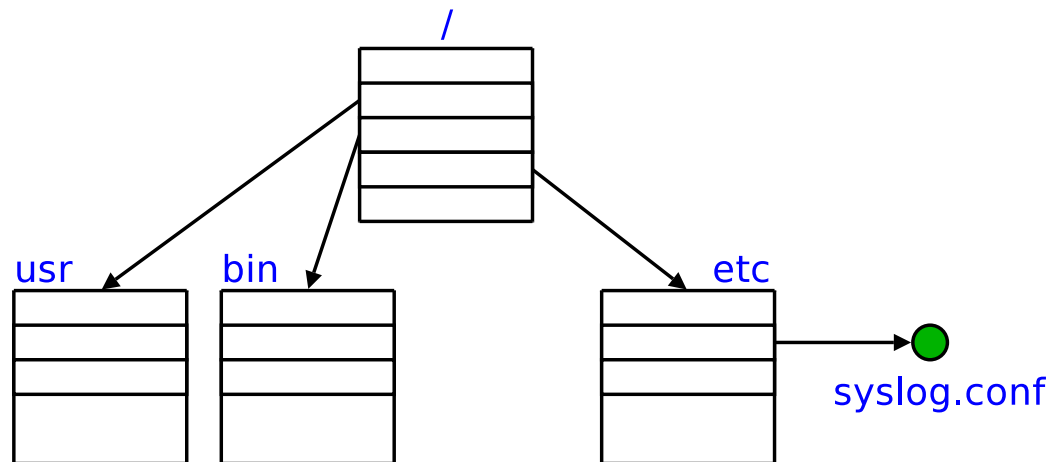
# Sistemas de Arquivos



Fonte: [1]

# Sistemas de Arquivos

- Ex: Caminho para o arquivo `syslog.conf` que fica dentro do diretório `etc`, que por sua vez está na raiz do sistema:
  - `/etc/syslog.conf`

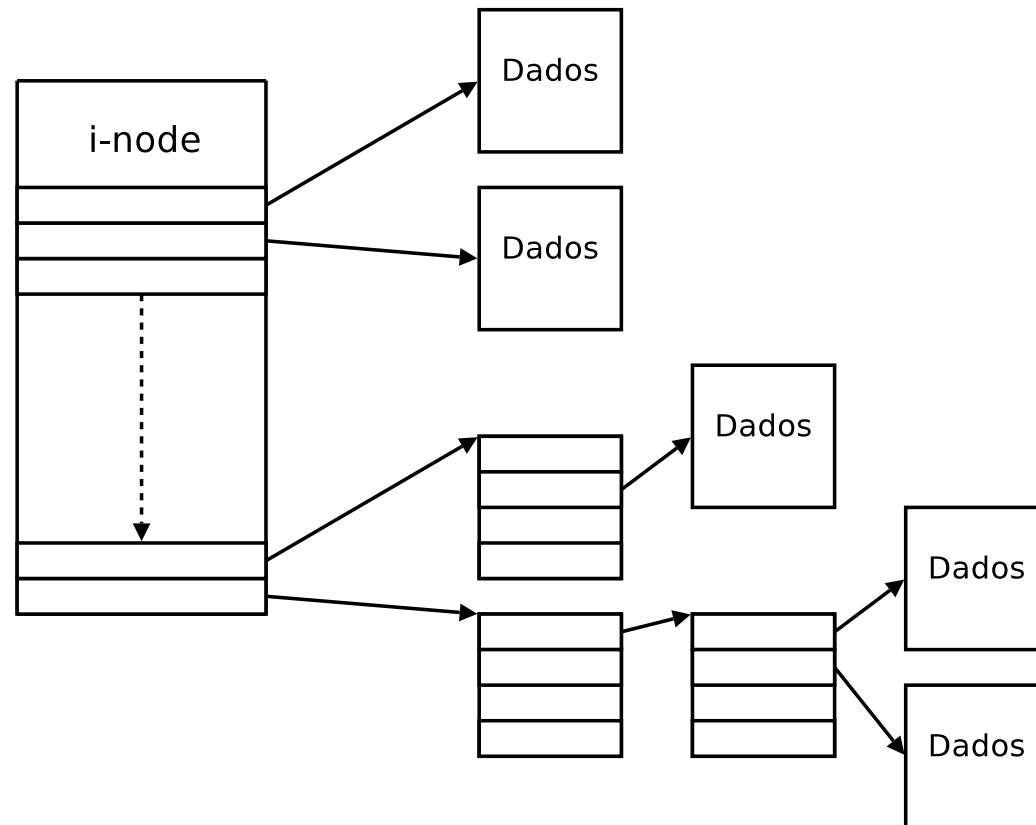


# Sistemas de Arquivos

- Sistema baseados em i-nodes
  - i-node é o *nó de índice* que aponta para os blocos no disco que contém os dados do arquivo, ou que contém endereços de blocos de dados do arquivo (endereçamento indireto).

# Sistemas de Arquivos

- Sistema baseados em i-nodes

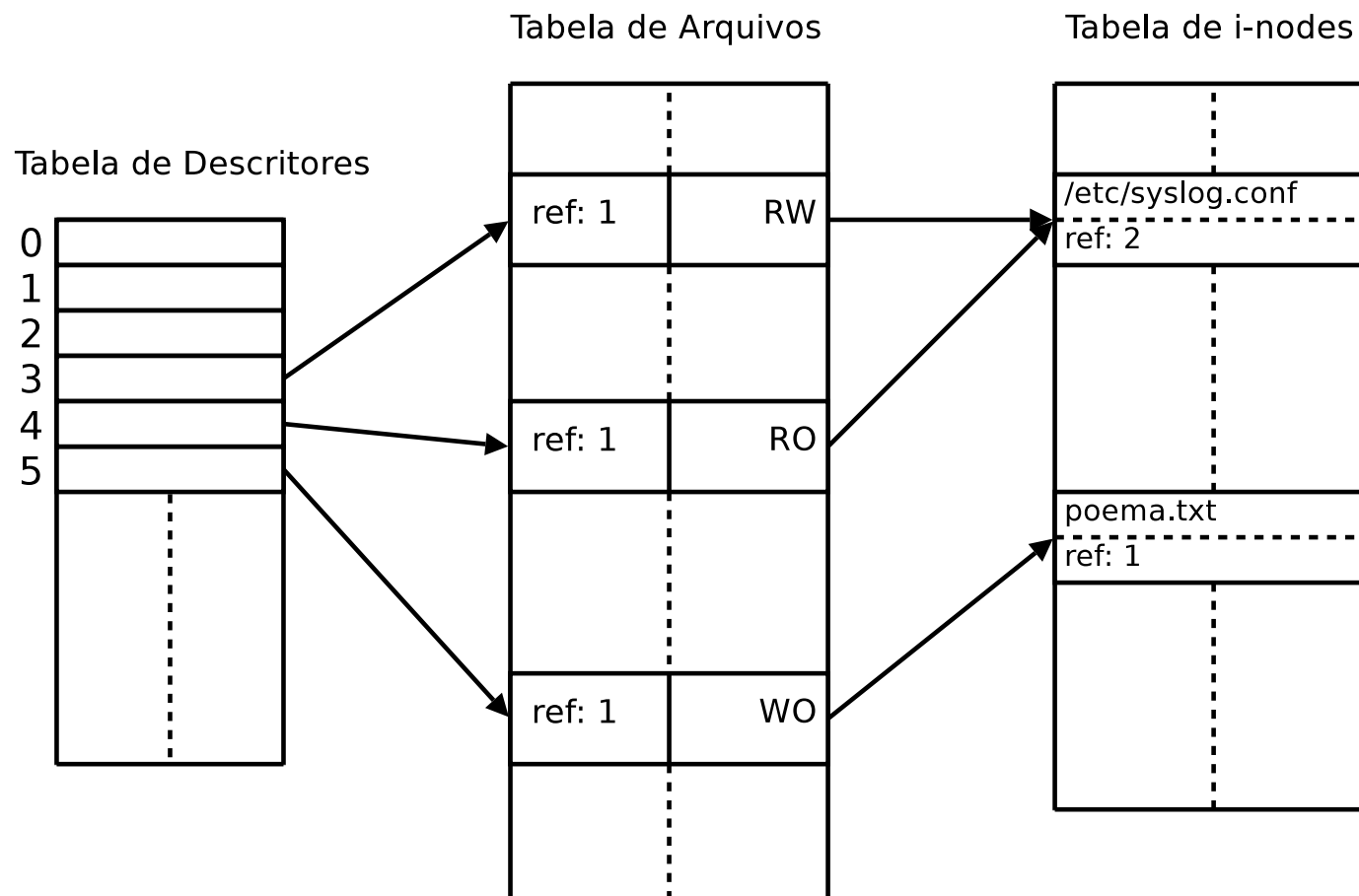


# Sistemas de Arquivos

- Sistema baseados em i-nodes
  - O Kernel possui uma tabela de arquivos abertos e uma tabela de i-nodes (referentes aos arquivos abertos), um arquivo pode ser aberto várias vezes ao mesmo tempo, mas apenas 1 i-node irá referenciá-lo.

# Sistemas de Arquivos

- Sistema baseados em i-nodes



# Sistemas de Arquivos

- O S.O. fornece *syscalls* para manipularmos arquivos, as principais são ***open***, ***read***, ***write***, ***lseek*** e ***close***:
  - `int open(const char *pathname, int flags, mode_t mode)`
  - `ssize_t read(int fd, void *buf, size_t count)`
  - `ssize_t write(int fd, const void *buf, size_t count)`
  - `off_t lseek(int fd, off_t offset, int whence)`
  - `int close(int fd)`

# open

● `int open(const char *pathname, int flags, mode_t mode)`

Função: Abre o arquivo fornecido

Entrada:

- `pathname`: Caminho do arquivo a ser aberto, ex: `/etc/syslog.conf`
- `flags`: Especifica como o arquivo será aberto: Somente para leitura (`O_RDONLY`), escrita (`O_WRONLY`), ambos (`O_RDWR`), truncado, etc
- `mode`: Especifica as permissões caso o arquivo seja criado, ex: `S_IRWXU`, `S_IRUSR`, `S_IWUSR`, `S_IXUSR`

Retorno:

- `int`: Índice para o descritor correspondente ao arquivo na tabela de descritores

# open

## ● Exemplo:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

/* ... */

int fd;

if((fd = open("/etc/syslog.conf", O_RDONLY)) < 0) {
    perror("\nErro ao abrir arquivo!\n");
} else {
    printf("\nArquivo aberto, descritor: %d\n", fd);
}

/* ... */
```

# open

- Note que no exemplo anterior, *open* foi chamada com apenas dois argumentos, apesar de ser declarada com 3 parâmetros. Por que o código funciona?

# open

- Note que no exemplo anterior, *open* foi chamada com apenas dois argumentos, apesar de ser declarada com 3 parâmetros. Por que o código funciona?
- O terceiro parâmetro (*mode*) só é utilizado quando utilizamos `O_CREAT` em *flags*, assim será empilhado lixo de memória como terceiro argumento, mas não há problemas, uma vez que não será utilizado.

# read

● `ssize_t read(int fd, void *buf, size_t count)`

Função: Lê dados do arquivo

Entrada:

- `fd`: Descritor do arquivo aberto
- `buf`: Ponteiro para o local em que os dados lidos serão armazenados (e necessário que `buf` tenha espaço suficiente)
- `count`: Quantidade de bytes a serem lidos

Retorno:

- `int`: -1 se houve erro, 0 se o encontrou final do arquivo ou o número de bytes lidos

# read

## ● Exemplo:

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

/* ... */

ssize_t n;
char buffer[1024];

n = read(fd, buffer, 1024);
if(n < 0) {
    perror("\nErro ao ler arquivo!\n");
} else {
    printf("\nBytes lidos: %d\n", n);
}

/* ... */
```

# write

● `ssize_t write(int fd, const void *buf, size_t count`

Função: Grava dados no arquivo

Entrada:

- `fd`: Descritor do arquivo aberto
- `buf`: Ponteiro para o local que contém os dados a serem gravados
- `count`: Quantidade de bytes a serem gravados

Retorno:

- `int`: -1 se houve erro, 0 se nada foi gravado ou o número de bytes gravados

# write

## ● Exemplo:

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

/* ... */

ssize_t n;
char buffer[] = "Ola Mundo!";

n = write(fd, buffer, strlen(buffer));
if(n < 0) {
    perror("\nErro ao gravar arquivo!\n");
} else {
    printf("\nBytes gravados: %ld\n", n);
}

/* ... */
```

# lseek

● `off_t lseek(int fd, off_t offset, int whence)`

Função: Seleciona o deslocamento do arquivo aberto

Entrada:

- `fd`: Descritor do arquivo aberto
- `offset`: Deslocamento
- `whence`: `SEEK_SET` desloca a partir do início do arquivo, `SEEK_CUR` a partir do deslocamento atual e `SEEK_END` a partir do final do arquivo

Retorno:

- `int`: Deslocamento atual ou -1 se houve erro

# lseek

- Simplificando, *lseek* seleciona a posição dentro do arquivo, assim podemos utilizar as funções *read* e *write* para lermos e escrevermos em qualquer ponto do arquivo, e não apenas sequencialmente.

# close

● `int close(int fd)`

Função: Fechar um arquivo aberto

Entrada:

- `fd`: Descritor do arquivo aberto

Retorno:

- `int`: -1 se houve erro, 0 se o arquivo foi fechado com sucesso

# close

## ● Exemplo:

```
#include <unistd.h>

/* ... */

/* Sem segredos */

close(fd);

/* ... */
```

# close

- Tudo que *close* faz é remover a entrada correspondente ao arquivo aberto da tabela de arquivos e decrementar a referência do *i-node* correspondente ao arquivo, se a referência for 0, então a entrada correspondente na tabela de *i-nodes* também é removida

# errno

- Nas *syscalls* vistas anteriormente, quando um erro ocorre é retornado o valor -1, e também é setada a variável global *errno* com o erro correspondente. A lista completa de erros listados por *errno* pode ser obtida nas páginas de manual do sistema.



**UFA! Chega de teoria, vamos praticar!**

# Pratica 1

- Objetivo:
  - Familiarizar o aluno com o ambiente de programação (gcc, make, etc) e com as chamadas de sistema (*syscalls*) através do uso da biblioteca C.

# Pratica 1

- Objetivo:
  - Familiarizar o aluno com o ambiente de programação (gcc, make, etc) e com as chamadas de sistema (*syscalls*) através do uso da biblioteca C.
- Faça um programa simples que utilize **todas** as chamadas vistas na aula de hoje: *open*, *read*, *write*, *lseek* e *close*



**Bom Trabalho!**

## Referências

- [1] Tanenbaum, A. S./ Woodhull A. S.; Sistemas Operacionais: projeto e implementação
- [2] Linux manual pages
- [3] Wikipedia, MS-DOS: <http://en.wikipedia.org/wiki/MS-DOS>
- [4] DOS INT 21h - DOS Function Codes  
<http://www.ee.iitm.ac.in/~ee01b044/interrupts.html>
- [5] Wikipedia, Microsoft Windows:  
[http://en.wikipedia.org/wiki/Microsoft\\_Windows](http://en.wikipedia.org/wiki/Microsoft_Windows)
- [6] Wikipedia, Linux: <http://en.wikipedia.org/wiki/Linux>

# Licença



Este documento é licenciado sob a  
Creative Commons Atribuição-Usó Não-Comercial 2.5  
Brasil License.