Insper

Sistemas Hardware-Software

Aula 15 - Entrada e Saída

Engenharia
Fabio Lubacheski
Maciel C. Vidal
Igor Montagner
Fábio Ayres

Correção

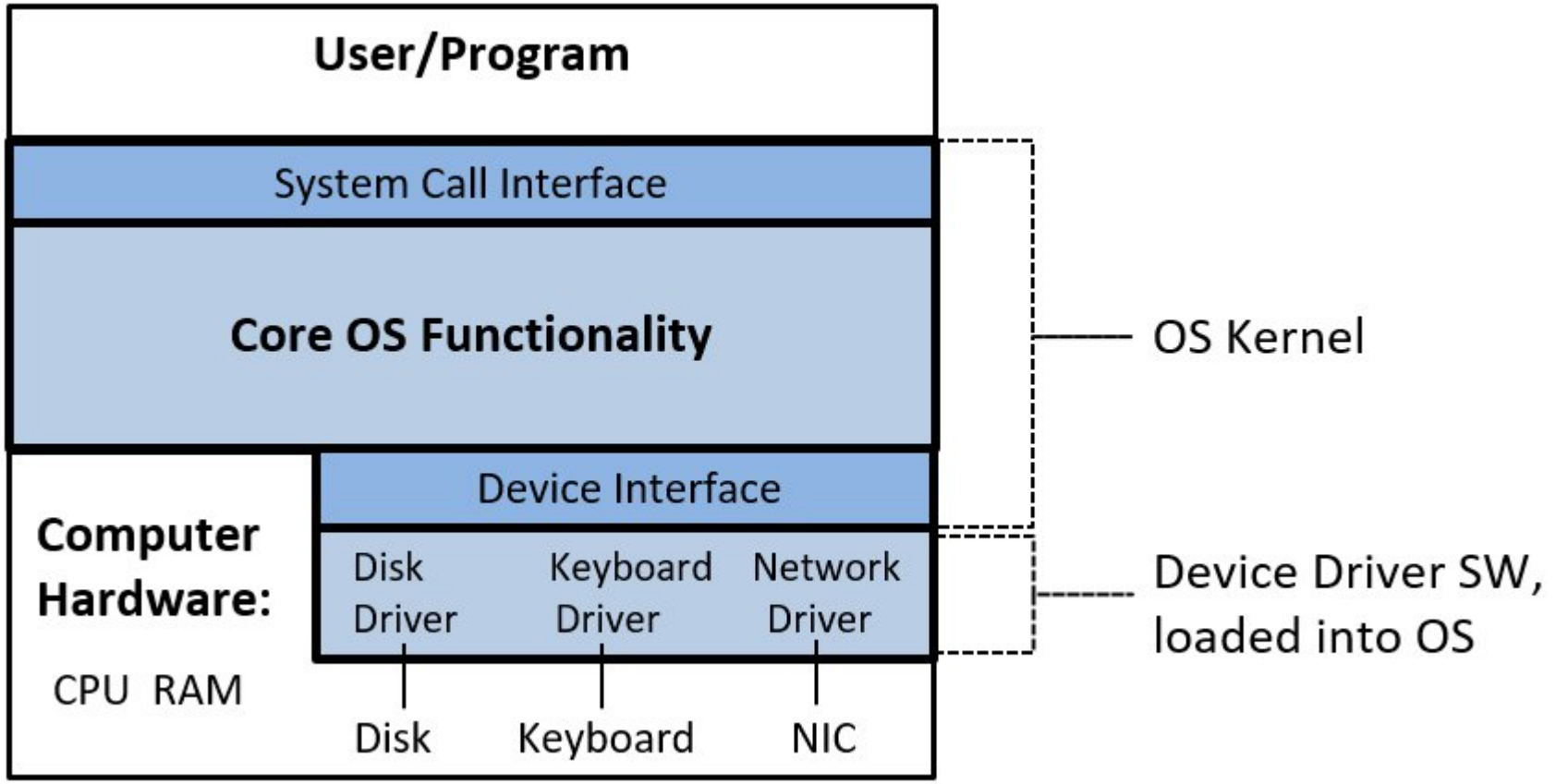
Exercício eh_par

1. Junção de fork + exec + wait

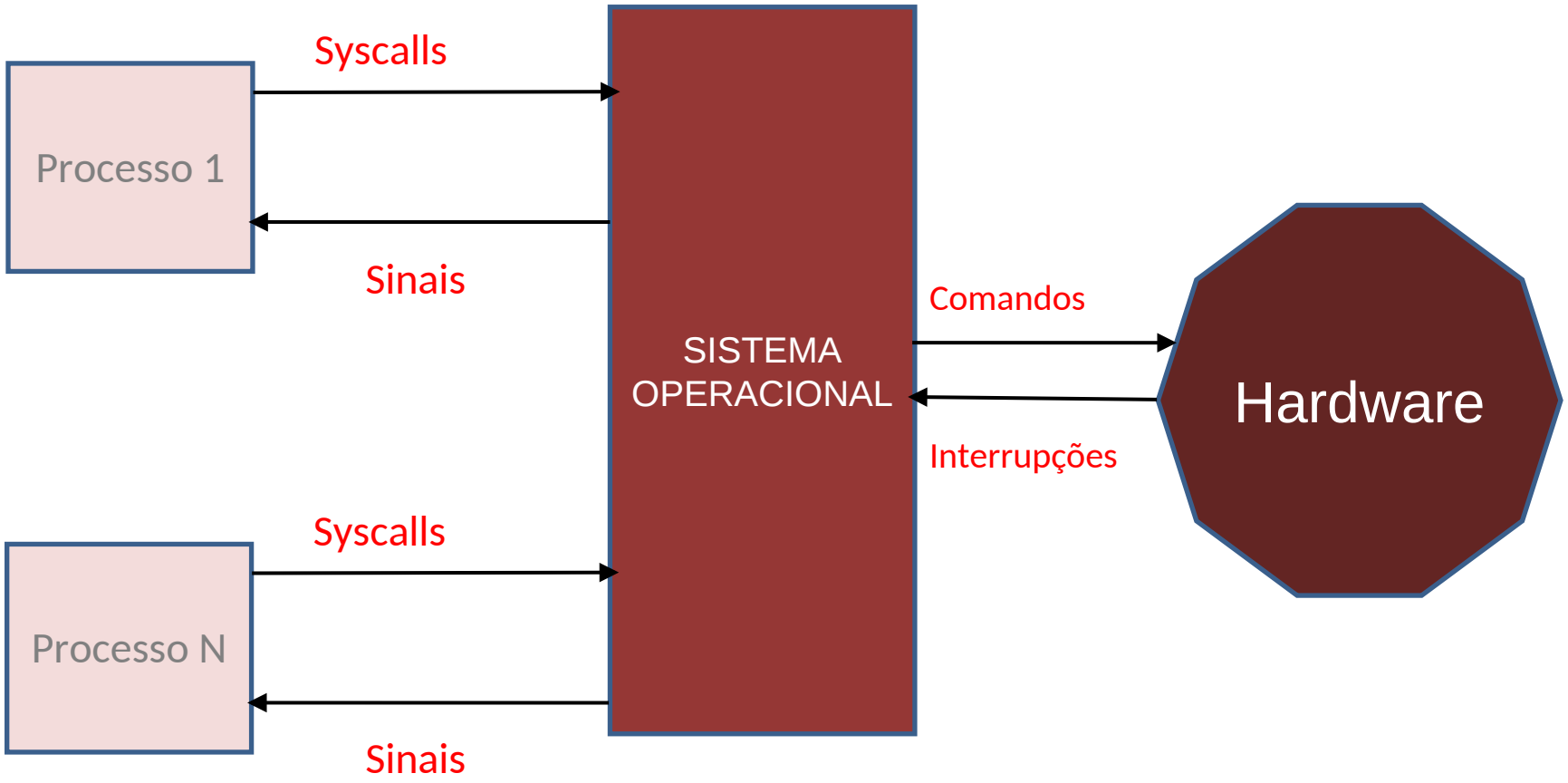
Objetivos de hoje

- Compreender o mecanismo usado pelo Sistema Operacional para expor **recursos de hardware**
- Utilizar chamadas de sistema POSIX para **ler** e **escrever arquivos**
- Compreender **permissões** de **arquivos** em sistemas POSIX

Estrutura de um Sistema Operacional



Sistemas Operacionais



Insper

SisCalls no Padrão POSIX

- Gerenciamento de usuários e grupos
- Manipulação de arquivos (incluindo permissões) e diretórios
- Criação de processos e carregamento de programas
- Comunicação entre processos
- Interação direta com hardware (via drivers)

Syscalls para arquivos

- Abrir e fechar arquivos: **open()** e **close()**
- Ler e escrever em arquivos: **read()** e **write()**
- Mudar posição corrente no arquivo: **lseek()**

Abrindo arquivos

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

- Retorna um inteiro chamado ***file descriptor***.
- **`flags`** indicam opções de abertura de arquivo
  - `O_RDONLY, O_WRONLY, O_RDWR`
  - `O_CREAT` (cria se não existir)
  - `O_EXCL + O_CREAT` (se o arquivo existir **open** falha e não cria o arquivo, ou seja, só cria o arquivo se não existir)
- **`mode`** define as permissões de um arquivo criado usando **open**.

  Ex: **0644**: Dono pode ler e escrever, outros só ler

Lendo/escrevendo em um arquivo

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

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

- Cada chamadas lê/escreve no máximo `count` bytes apontados por **`buf`** no arquivo **`fd`**.
- Ambas retornam o número de bytes lidos/escritos e **`-1`** se houver erro.
- Se **`read`** retornar **`0`** se acabou o arquivo.

Exemplo usando arquivo – ex1.slides.c

```c
int main () {
    int fd;
    char c;
    // abre o arquivo, retorna -1
    // se o arquivo não existir
    fd = open("entrada.txt", O_RDONLY);
    // le um caractere do arquivo
    read(fd,&c,1);

    printf("c = %c\n", c);
    // Fechar um arquivo informa ao kernel
    // que você já terminou de acessar
    // o arquivo. SEMPRE FAÇA ISSO.
    close(fd);
    return 0;
}
```

Tipos de arquivos

- **Arquivos regulares**
  - Dados arbitrários. Ex: texto, binários, código-fonte, imagens, etc

- **Diretórios**
  - Um índice para um **arquivo especial** que contém uma **lista de nomes de arquivos**.

    Ex: `/home/usuario/,/etc/, ./minhapasta/`

- **Sockets**
  - Para comunicar com outro processo em outra máquina

Arquivos regulares

Para o kernel, não existe diferença entre “arquivo texto” e “arquivo binário”: é tudo byte!

Arquivos texto: conceitualmente são uma sequência de linhas

Término de linhas:

- Linux e MacOS: *newline* ou *line feed* (‘\n’)
- Windows e protocolos Internet: *carriage return* seguido de *line feed* (‘\r\n’)

  \r – ASCII 13

  \n – ASCII 10

Diretórios

Um diretório é um array de links, mapeando um nome de arquivo a um arquivo

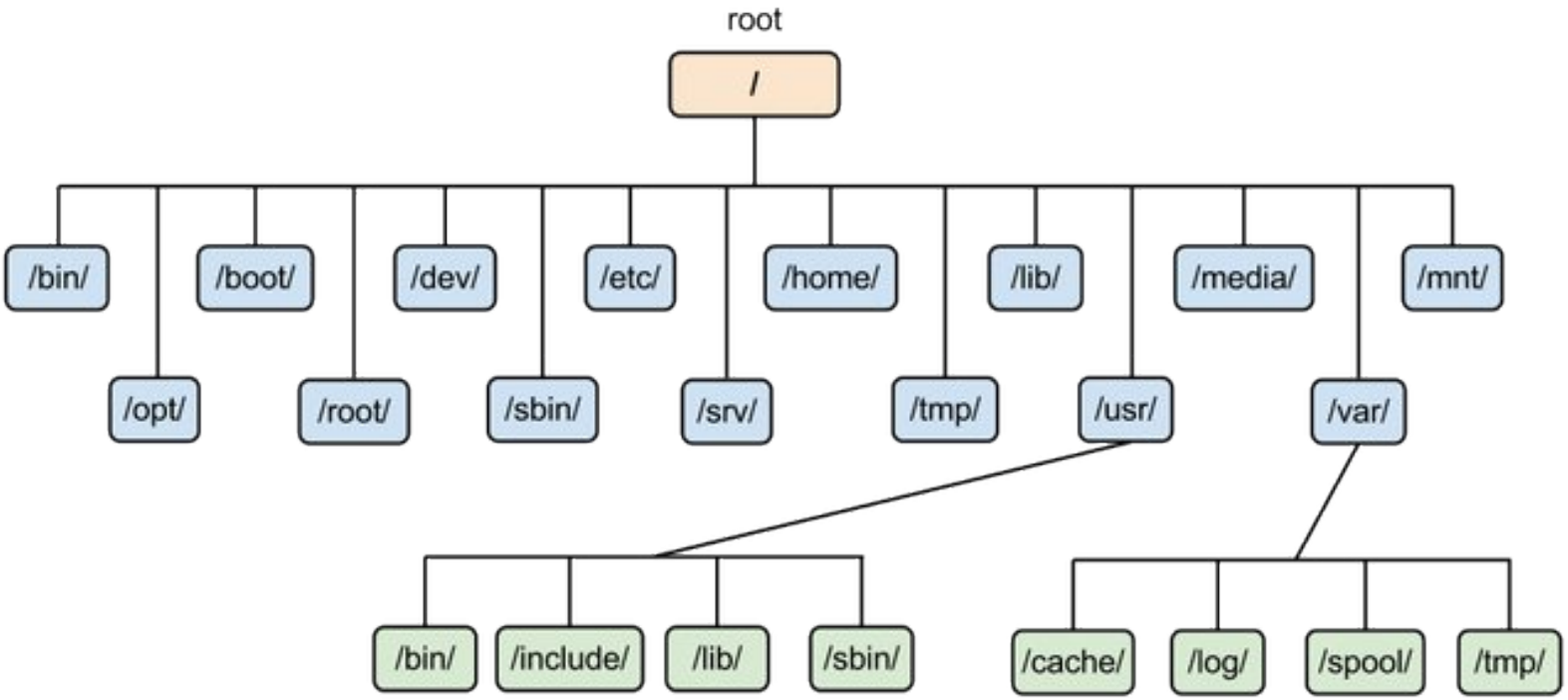
Todo diretório contém ao menos duas entradas:

- `.` (dot) é um link para si próprio
- `..` (dot dot) é um link para o diretório pai na hierarquia de diretórios

Comandos: `mkdir`, `ls`, `rmdir`

Cada processo roda em um diretório corrente (current working directory – `cwd`), que pode ser alterado com `chdir()`

Arquivos em Unix




```
$man hier
```

Fonte: https://nepalisupport.wordpress.com/2016/06/29/linux-file-system-hierarchy/

Permissões de arquivos

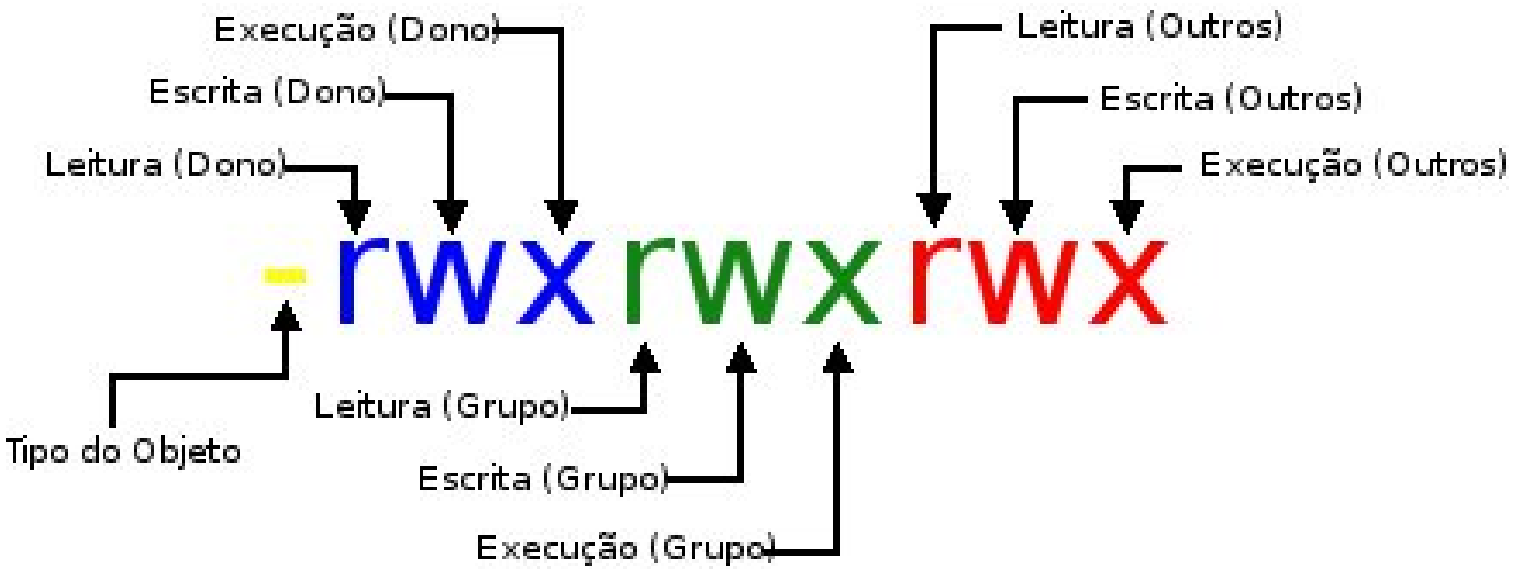
- Funcionam dentro da camada de Aplicação.
- Cada arquivo possui um usuário dono
- Permissões de leitura(4), escrita(2) e execução(1) para
  - Usuário dono do arquivo
  - Usuários no mesmo grupo de usuários do dono
  - Todo mundo
- Permissões codificadas usando números de 0 a 7

```
$man chmod
$man chown
```

Visualizando permissões no Linux

- Para visualizar a permissão de um diretório, podemos ir até o local onde esse diretório se encontra. E digitar ls -l (lista com detalhes os arquivos

```
$ ls -l entrada.txt
-rw-r--r-- 1 fabio fabio 2 out 23 09:34 entrada.txt
```



Metadados – informação de um arquivo

```
/* Metadata returned by the stat and fstat functions */
struct stat {
    dev_t          st_dev;      /* Device */
    ino_t          st_ino;      /* inode */
    mode_t         st_mode;     /* Protection and file type */
    nlink_t        st_nlink;    /* Number of hard links */
    uid_t          st_uid;      /* User ID of owner */
    gid_t          st_gid;      /* Group ID of owner */
    dev_t          st_rdev;     /* Device type (if inode device) */
    off_t          st_size;     /* Total size, in bytes */
    unsigned long  st_blksize;  /* Blocksize for filesystem I/O */
    unsigned long  st_blocks;   /* Number of blocks allocated */
    time_t         st_atime;    /* Time of last access */
    time_t         st_mtime;    /* Time of last modification */
    time_t         st_ctime;    /* Time of last change */
};
```

Descrição detalhada em **`man 2 stat`**

Atividade prática

Trabalhando com arquivos (40 minutos)

1. Usar funções básicas de tratamento de arquivos

Atividade prática

Permissões e posse de arquivos (30 minutos)

1. Utilizar e entender as permissões de arquivo nas funções de E/S

Insper

www.insper.edu.br