Insper

Sistemas Hardware-Software

Aula 21 – Entrada e saída II

2019 – Engenharia

Igor Montagner, Fábio Ayres <igorsm1@insper.edu.br>

Aulas passadas

- Entrada e saída
  - `open, close`
  - `read, write`
  - `lseek`
- Permissões e sistema de arquivos
  - `chmod, chown`
  - `sigprocmask e sigset_t`

Aulas passadas

- Criação e gerenciamento de processos
  - `fork`
  - `exec*`
  - `wait*`
  - `exit`
- Sinais
  - `kill/raise`
  - `sigaction/signal`
  - `sigprocmask e sigset_t`

Comentários sobre sinais

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_CREATE` (cria se não existir)
  - `O_EXCL + O_CREATE` (se existir falha)
- `mode` indica as permissões de um arquivo criado usando open.

E/S padrão

Todo processo criado por um shell Linux já vem com três arquivos abertos, e associados com o terminal:

0: standard input (stdin)

1: standard output (stdout)

2: standard error (stderr)

Fechando um arquivo

Fechar um arquivo informa ao kernel que você já terminou de acessar o arquivo.

```
int fd;      /* file descriptor */
int retval; /* return value */

if ((retval = close(fd)) < 0) {
   perror("close");
   exit(1);
}
```

Cuidado: não feche um arquivo já fechado!

Lendo/escrevendo em um arquivo

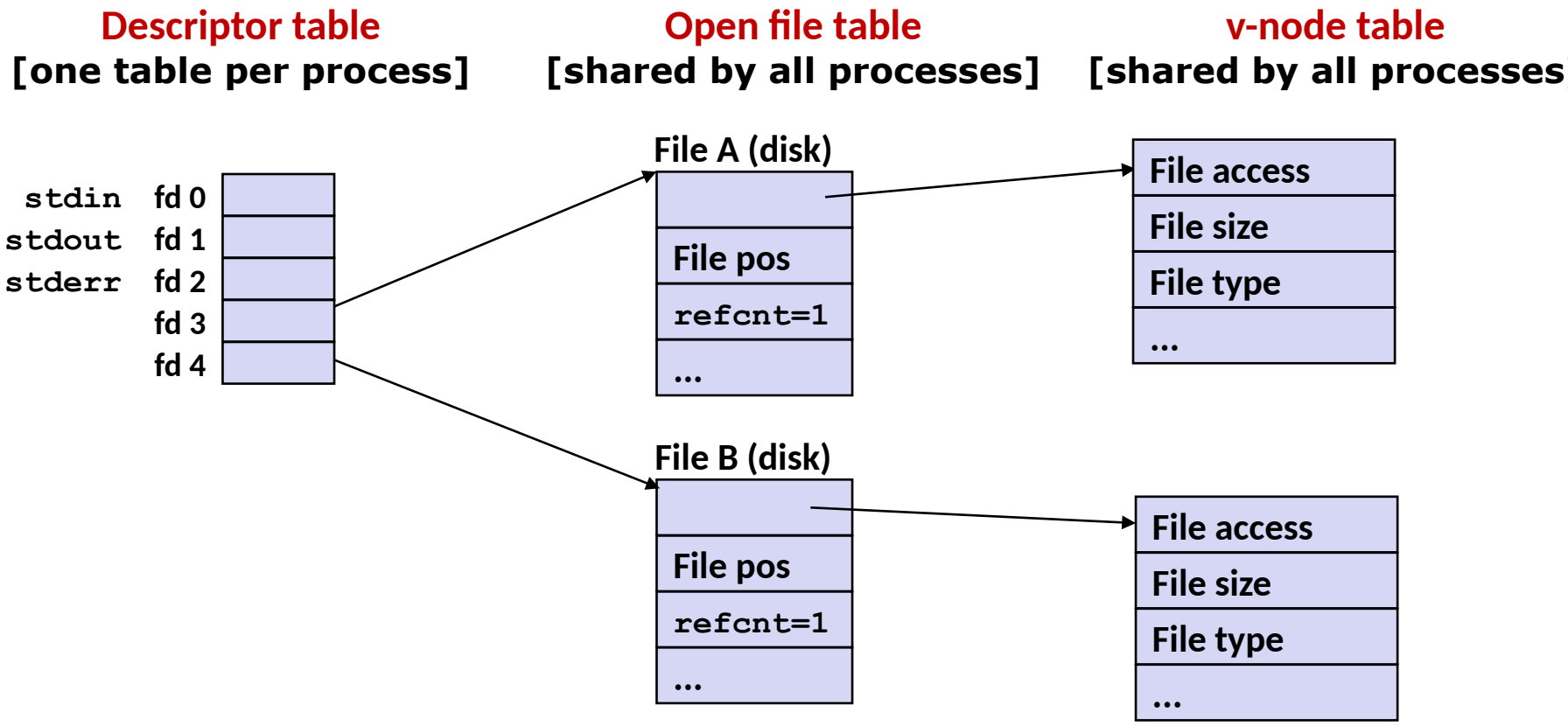
```
ssize_t read(int fd, void *buf, size_t count);

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 acabou o arquivo.

Hoje

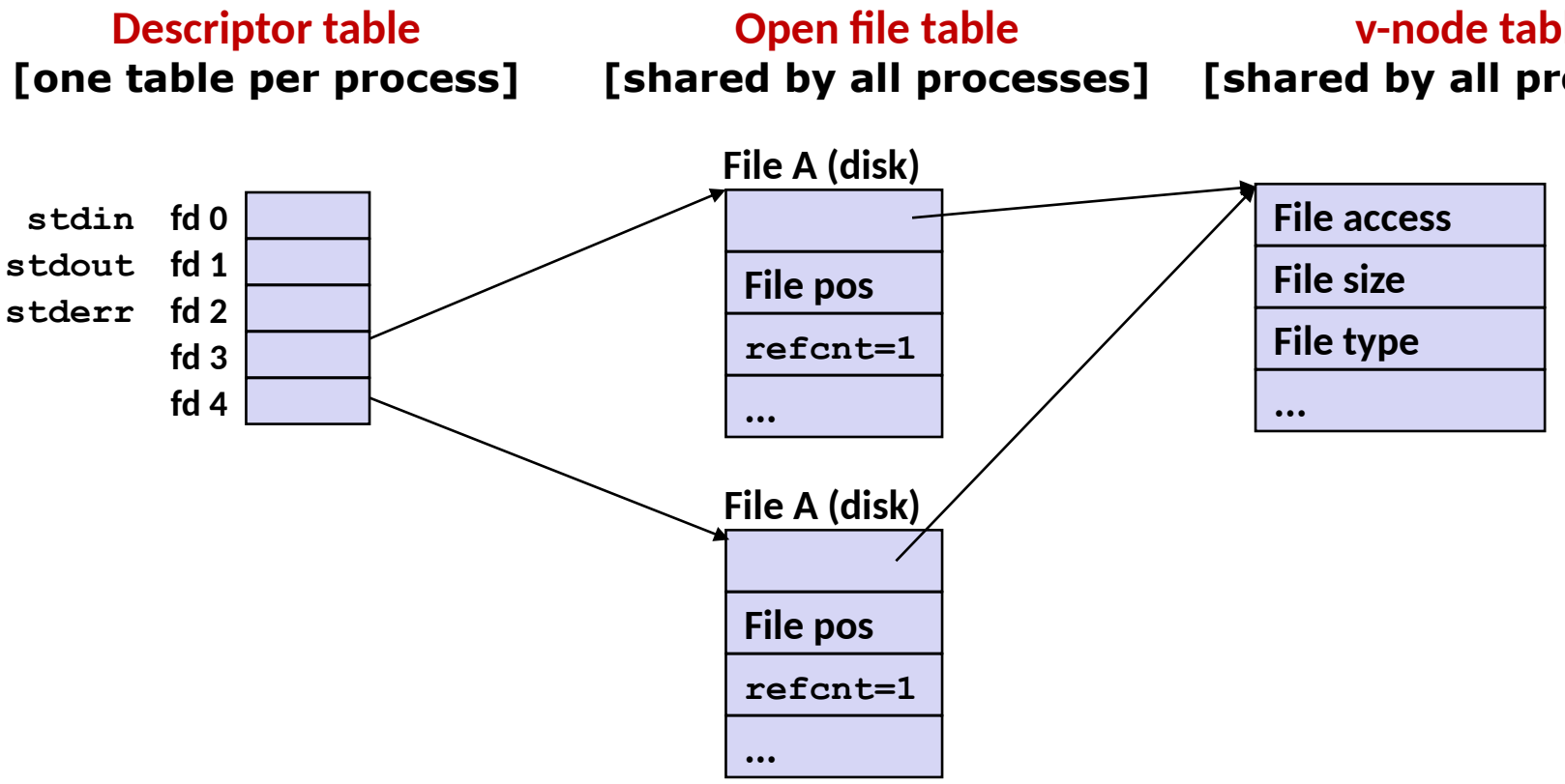
- Redirecionamento de arquivos
- Comunicação (simples) entre processos

Tabela de descritores de arquivos



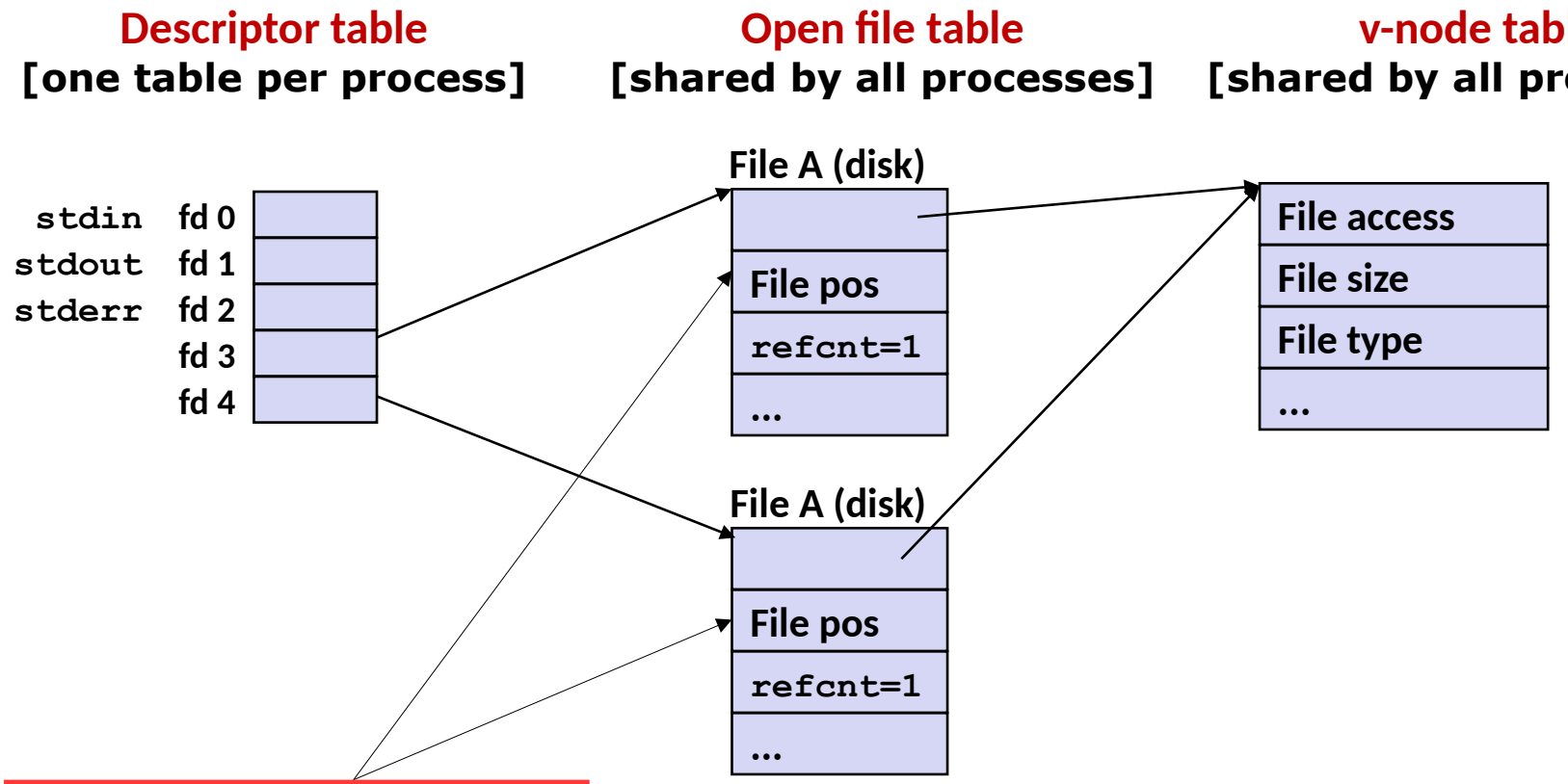
Insper

Tabela de descritores de arquivos



Insper

Tabela de descritores de arquivos

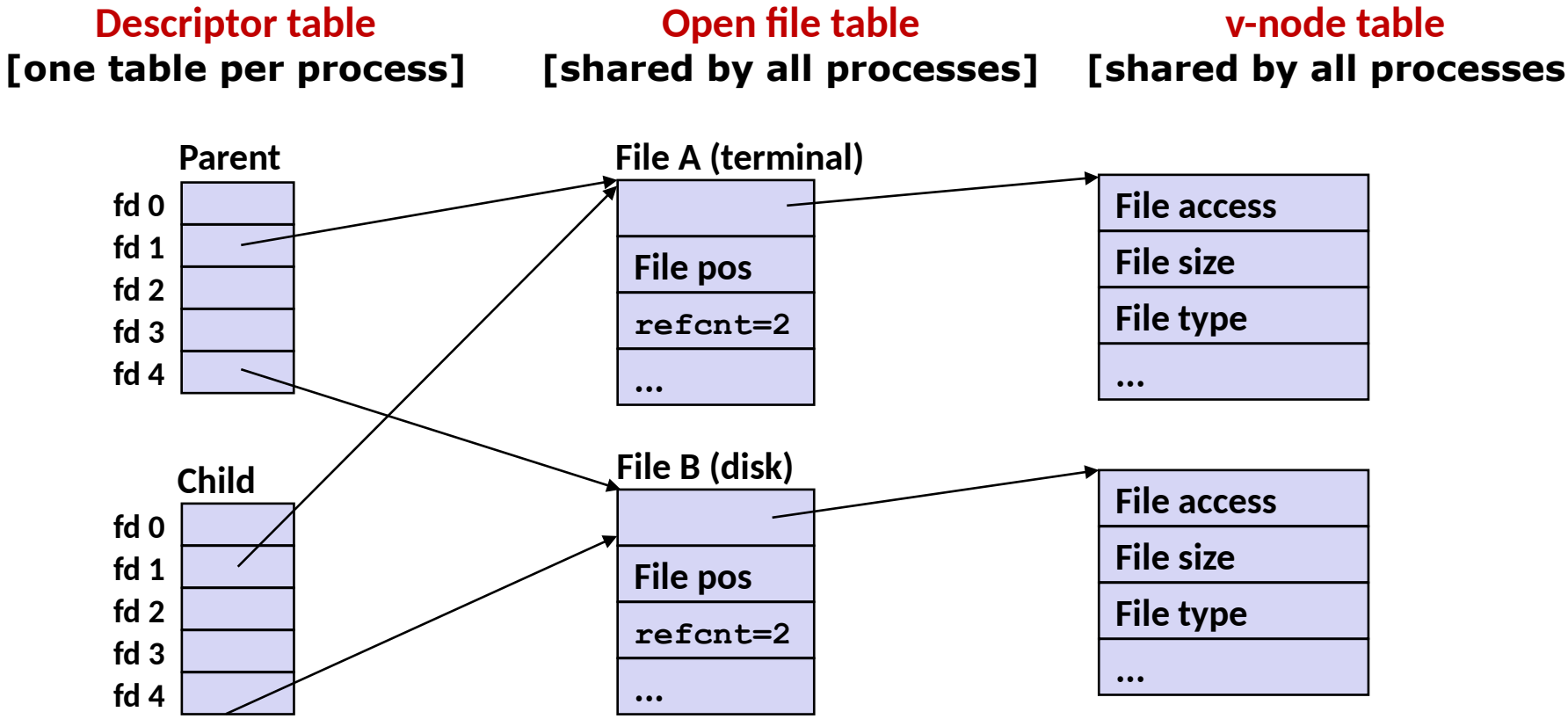


Cada um tem sua posição de leitura/escrita atual!

Insper

fork

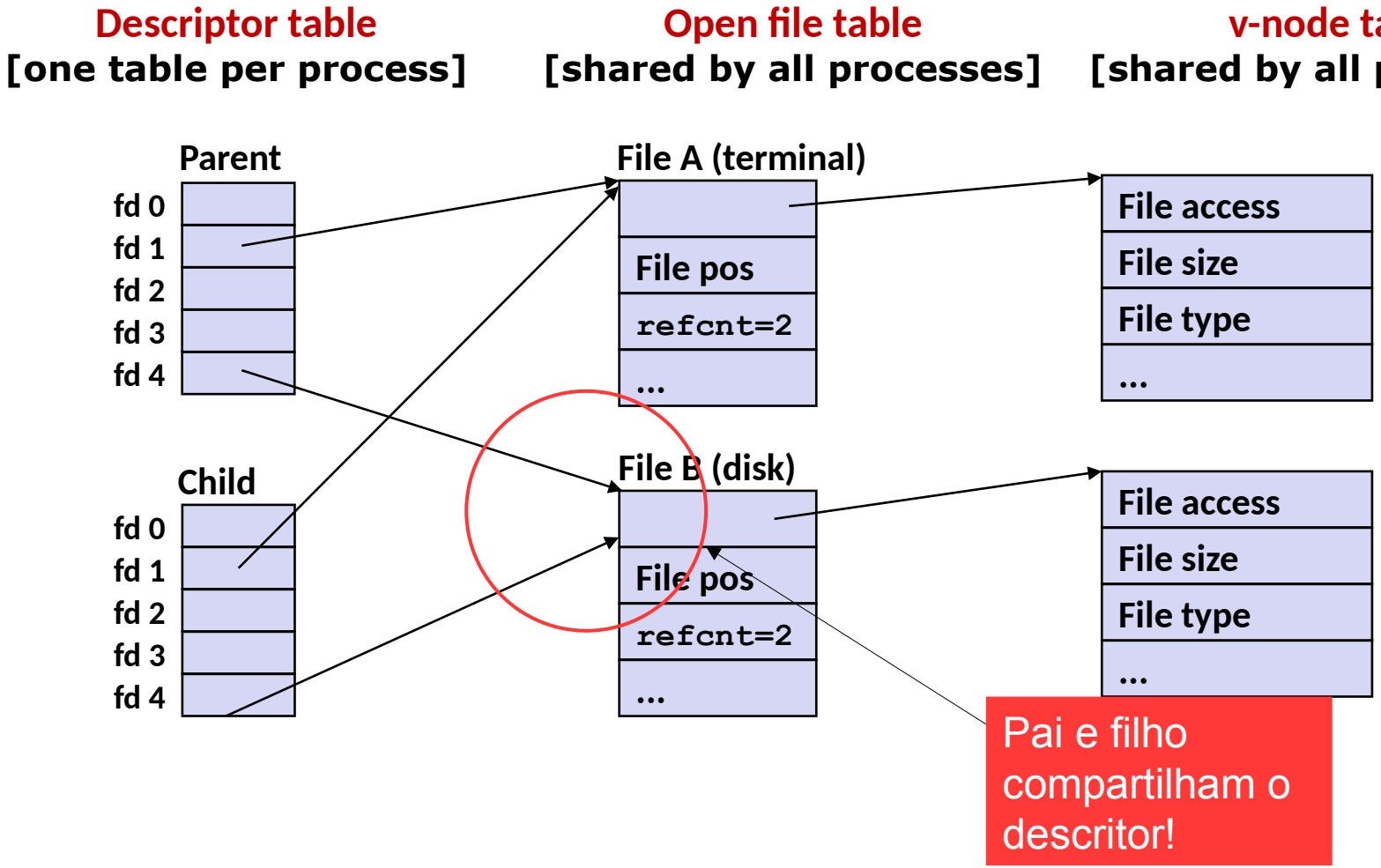
Depois do fork



Insper

fork

Depois do fork



Insper

Descritores de arquivos e fork

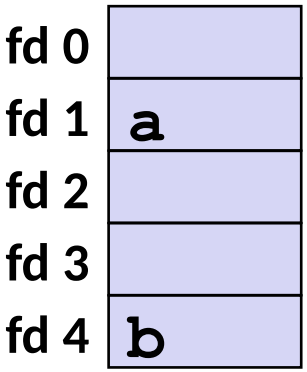
- São mantidos após o fork e exec
- Acessos podem ter problemas de concorrência
- Podemos manipular os descritores de processos filhos

Redirecionamento de I/O

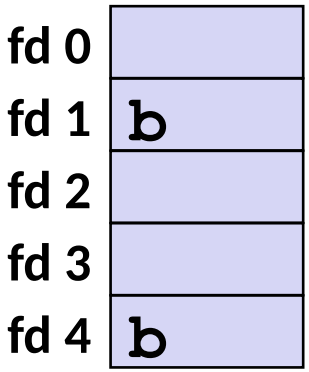
`dup2(oldfd, newfd)`

Copia o valor da posição **`oldfd`** para a posição **`newfd`** da tabela de descritores

Descriptor table
before **`dup2(4,1)`**



Descriptor table
after **`dup2(4,1)`**

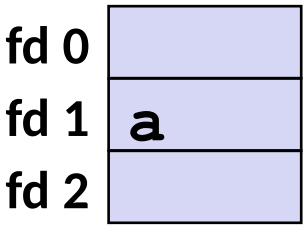


Redirecionamento de I/O

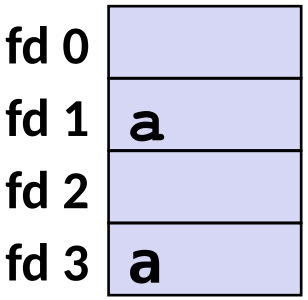
```
int d = dup(oldfd)
```

Cria um novo fd que aponta para o mesmo arquivo que oldfs

Descriptor table
before **`dup(1)`**



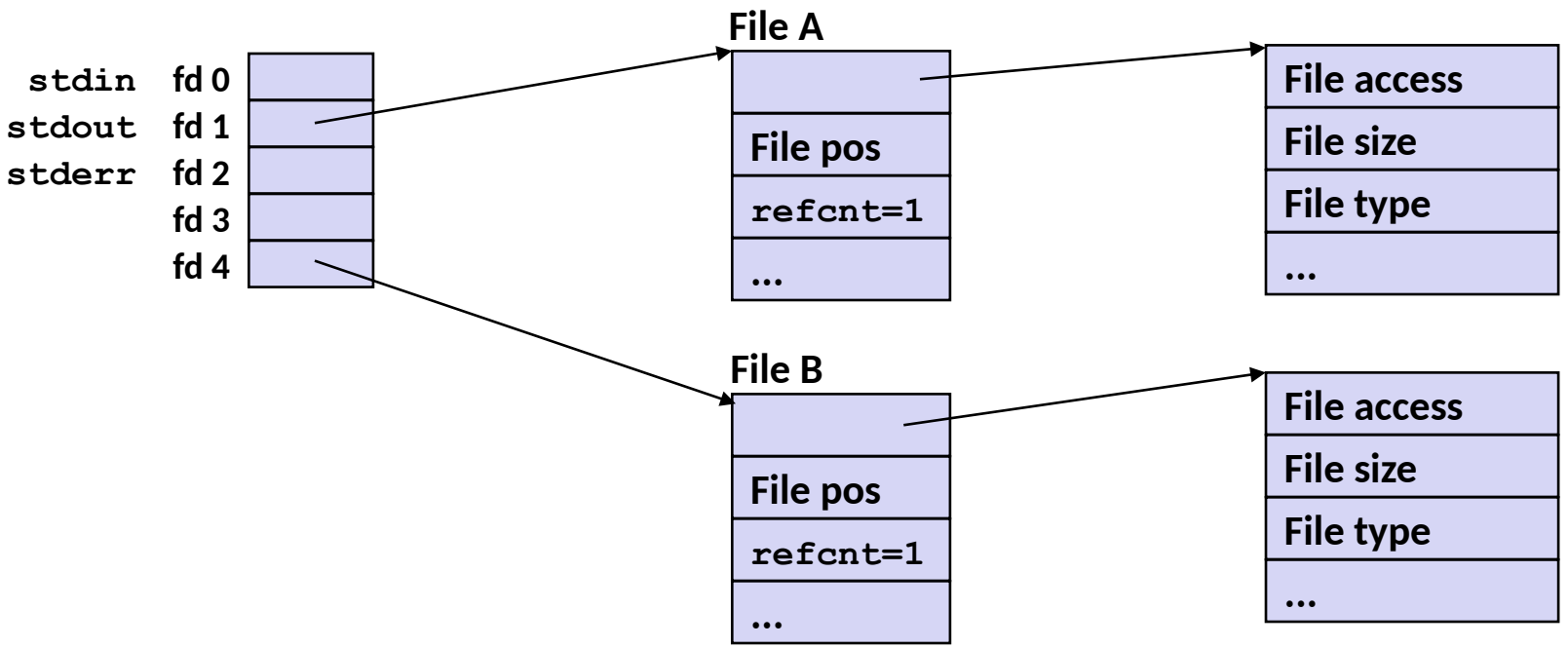
Descriptor table
after **`dup(1)`**



Insper

Redirecionamento de I/O

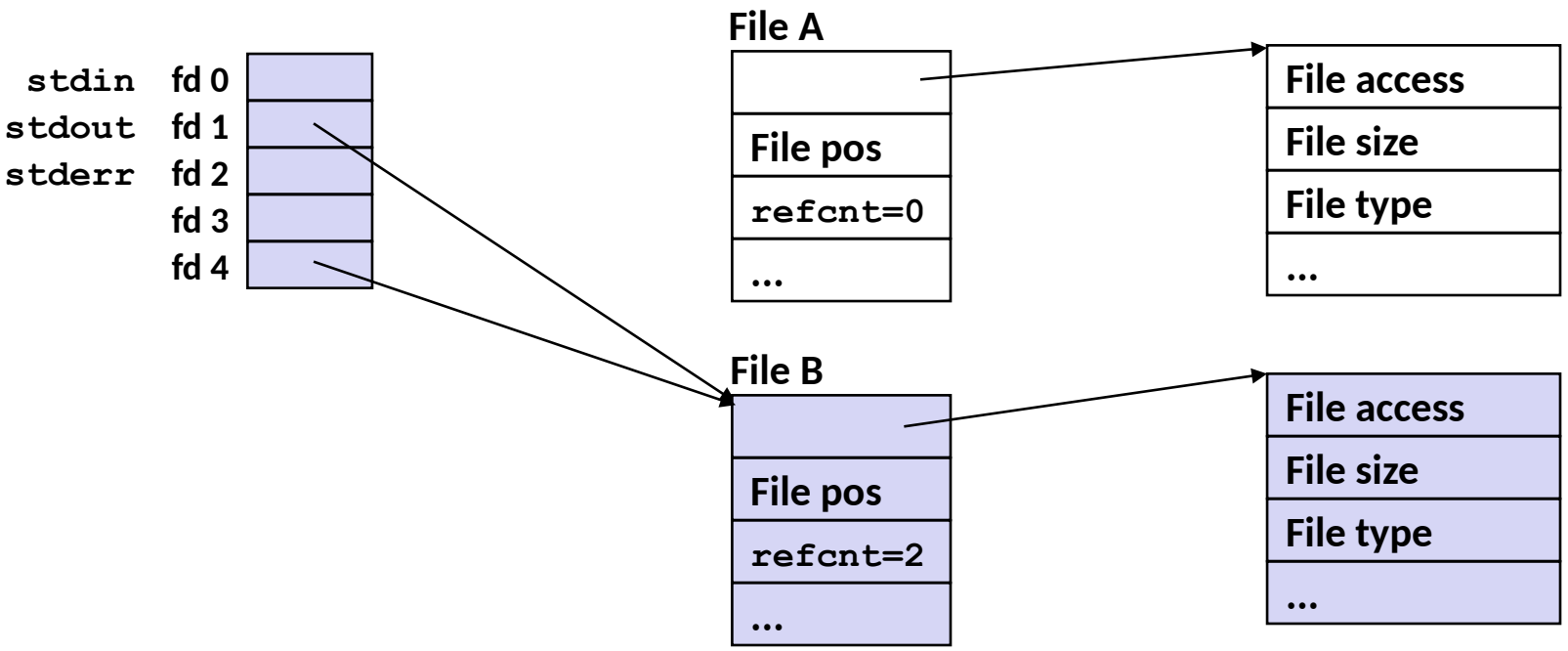
1. Abrir o arquivo para o qual stdout será redirecionado



Insper

Redirecionamento de I/O

2. Chamar dup2(4, 1)



Insper

Redirecionamento de I/O - usos

- Salvar saída um comando para arquivos
- Automatizar a digitação de comandos ao redirecionar a entrada de um programa
- Permitir a comunicação entre dois programas a partir da entrada/saída padrão

Redirecionamento de I/O – Atividade

- Parte 1 do roteiro de hoje – 30 minutos

Comunicação entre Processos (IPC)

Chamada **pipe** permite comunicação unidirecional entre processos

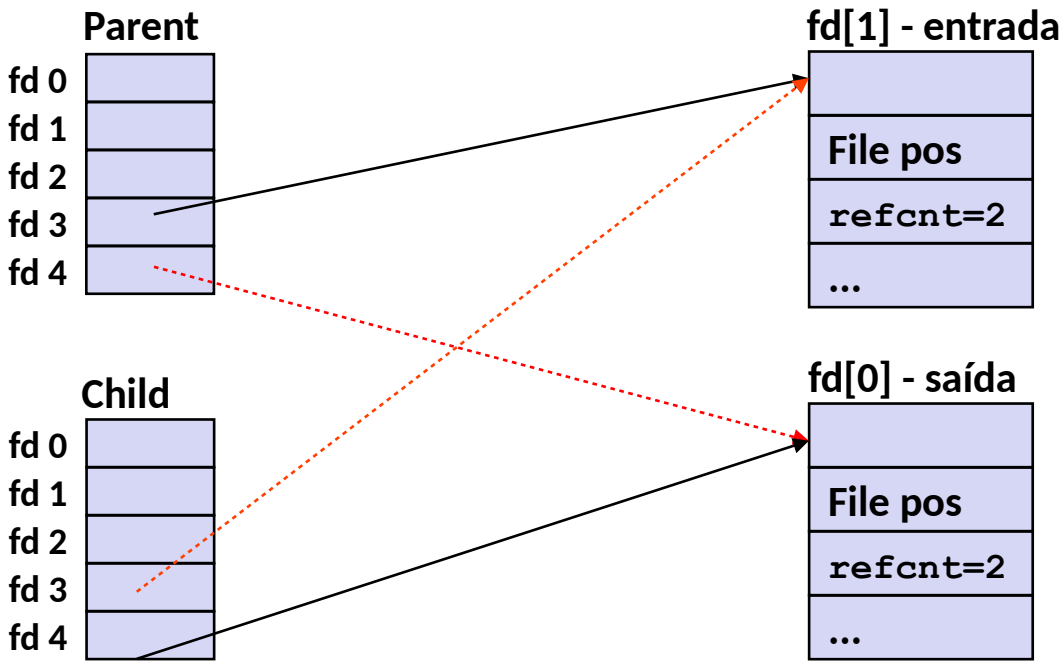
```
#include <unistd.h>
int pipe(int fd[2]);
```

Tudo o que for escrito (usando *write*) em *fd[1]* fica disponível para leitura em *fd[0]* (usando *read*)

Podemos ler e escrever nos arquivos a partir de diferentes processos. É possível enviar informações de um processo para o outros desta maneira.

Redirecionamento de I/O - II

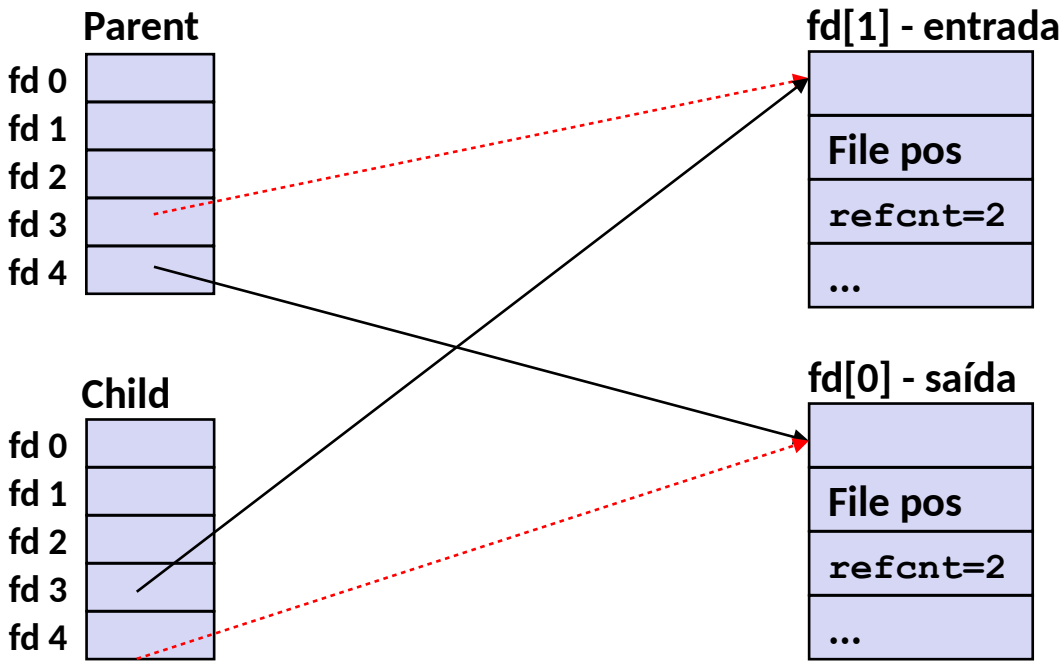
Comunicação unidirecional entre processos: Parent → Child



Insper

Redirecionamento de I/O - II

Comunicação unidirecional entre processos: Child → Parent



Insper

Limitações de pipes

```
#include <unistd.h>
int pipe(int fd[2]);
```

- Comunicação unidirecional
- Só vale para processos na própria máquina
- Precisa ocorrer antes do fork

Outras técnicas de IPC

- UNIX domain sockets (arquivo)
- Sockets (via rede) locais
- Memória compartilhada

Mais complexas, não veremos nesta disciplina

Pipes – Atividade

- Parte 2 do roteiro de hoje – 30 minutos

Insper

www.insper.edu.br