Anotações - Guião 1

#SoftwareEngineering #SO #C

Descritor de Ficheiro

Representado por um inteiro não negativo.
Representação abstrata de um ficheiro utilizada para operar sobre o mesmo

0 -> standard input
1 -> standard output
2 -> standard error

Estruturas Kernel

Tabela de processos (TP)

- Uma tabela por processo
- Guarda descritores de ficheiros abertos

Tabela de ficheiros (TF)

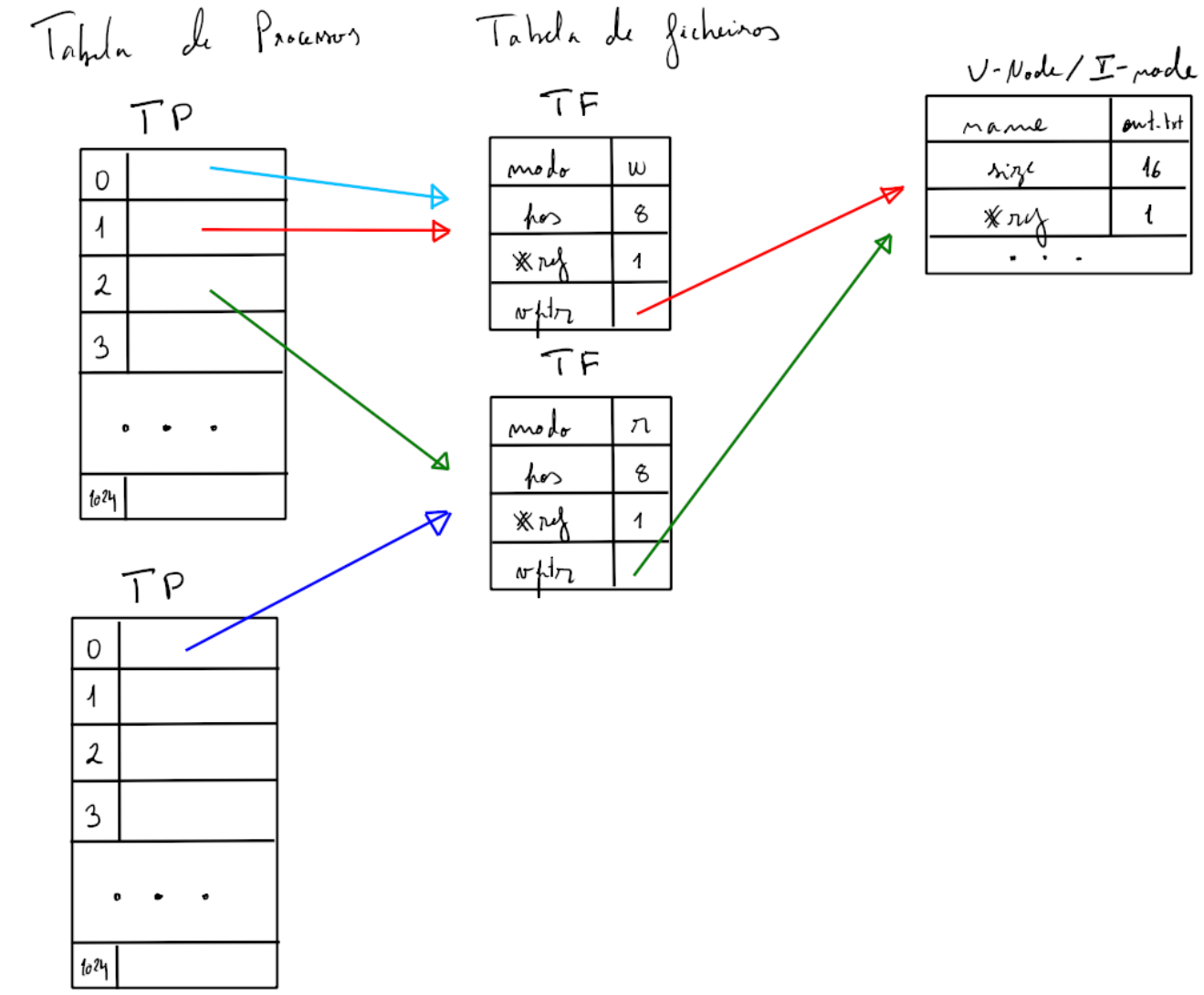
- Tabela partilhada pelo sistema operativo
- Guarda *modo de abertura* e *posição de leitura/ escrita* de cada descritor

V-node

- abstração de um objeto Kernel que respeita a interface de um ficheiro UNIX
- permite representar ficheiros, diretorias, FIFOs, *domain sockets*, ...
- guarda informação do tipo de objeto, apontadores para as funções sobre o mesmo e para o respetivo i-node

I-noded

- guarda metadados/atributos dos ficheiros (ex.: nome, tamanho, ...)
- guarda localização dos dados no recurso fisico de amazenamento



- Entradas na tabela de ficheiros de sistema podem partilhar inodes
- Descritores de **processos distintos** podem partilhar entradas na tabela de ficheiros de sistema
- Descritores do **mesmo processos** oidem partilhar entradas na tabela de ficheiros de sistema

Chamadas ao sistema

`<unistd.h>` - definições e declarações de chamadas
`<fcntl.h>` - definição modos de abertura de ficheiro

```
int open (const char *path, int oflag [, mode]);
```

- inicializa um *descritor* para um determinado ficheiro
- devolve um descritor ou erro
- **path** - caminho do ficheiro
- **oflag** - modo de abertura (O_RDNLY, O_WRONLY...)
- **mode** - permissões de acesso para O_CREAT
  - e.g. 0640 (octal) equivale a rw-r-----

As flags

- `O_RDONLY`: Abrir apenas para leitura
- `O_WRONLY`: Abrir apenas para escrever
- `O_RDWR`: Abrir para ler e para escrever

Indicam o modo de abertura e não podem ser misturadas umas com as outras, por exemplo, `open("file", O_RDONLY | O_WRONLY)` está errado.

As outras flags que podem ser passadas indicam opções.

- `O_CREAT`: Cria o ficheiro se ele não existir.
- `O_APPEND`: Começa a escrever no fim do ficheiro em vez de no início.
- `O_TRUNC`: Apaga todo o conteúdo do ficheiro.
- etc, consultar a man page `man 2 open` para saber mais.

O parâmetro opcional `permissions`, permite especificar as permissões com que o ficheiro deve ser criado (caso a flag `O_CREAT` seja especificada), este é o mesmo número octal que pode ser passado para o `chmod` para especificar as novas permissões do ficheiro.

Nota: Em C um número octal é começado por um `0`.

Por exemplo, para criar um ficheiro com permissões para leitura e escrita para o dono e só leitura para todos os outros utilizadores, chama-se o `open` da seguinte forma.

```
open("file", O_CREAT | O_WRONLY, 0644);
```

Nota: Se não forem especificadas as permissões quão é criado um ficheiro as permissões com que ele é criado não são especificadas, podendo ser qualquer coisa.

Valor de retorno

O valor de retorno pode ter 2 significados.

Se for:

- `≥ 0`: um *file descriptor* que pode mais tarde ser passado a varias funções para interagir com o ficheiro aberto
- -1: Indica que ocorreu algum erro

```
ssize_t read(int fildes, void *buf, size_t nbyte);
```

- devolve número de bytes lidos ou erro
- fildes - descritor ficheiro
- buf - buffer para onde conteúdo é lido
- nbyte - número max de bytes a ler *(buffer overrun?)*

Parâmetros

O `fd` é um *file descriptor* do ficheiro de onde ler.
O `buf` é um buffer/array para onde o conteúdo vai ser lido.
O `count` é quantos bytes no máximo devem ser lidos para dentro do buffer.

Valor de retorno

O valor de retorno pode ter 3 significados.

Se for

- `> 0`: Indica quantos bytes foram lidos
- 0: Indica que o ficheiro terminou
- -1: Indica que ocorreu algum erro

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

Parâmetros

O `fd` é um *file descriptor* de ficheiro para onde escrever.
O `buf` é o buffer/array onde ir buscar os dados que devem ser escritos.
O `count` é quantos bytes do `buf` devem ser escritos

Valor de retorno

O valor de retorno pode ter 2 significados.

Se for

- `≥ 0`: Indica quantos bytes foram escritos
- -1: Indica que ocorreu algum erro

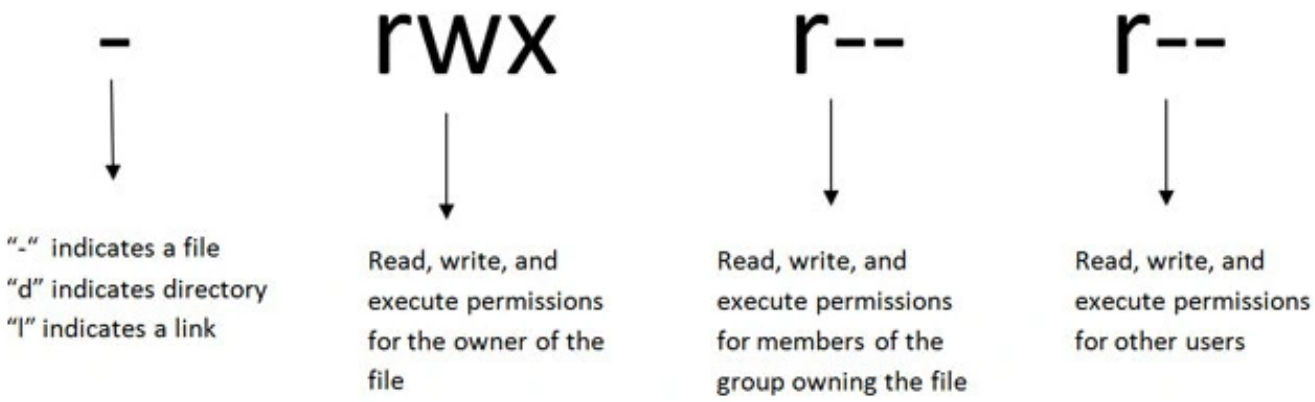
`int close(int fildes);`

- apaga o descritor da tabela do processo
- devolve 0 caso a operação

Posição (offset)

- A cada operação de leitura/ escrita efetuada sobre o mesmo descritor, a posição a ler/escrever é atualizada consoante o número
  - Exemplo: `` `int fd = open("foo.txt", O_CREAT | O_TRUNC | O_RDWR, 0600)` ``
    - Não esquecer as permissões corretas se usar `O_CREAT`: `RW => 06 ou 07`
- Situação de "End of File" na tentativa de leitura (ex.: `buf = 8, pos = 8 e size = 8`)
  - Para acesso sequencial, pode-se criar/ ter um outro descritor para leitura do mesmo ficheiro
  - Para uma solução mais geral de acesso direto read/ write deve-se recorrer à chamada `lseek`
    - `off_t lseek(int fildes, off_t offset, int whence);`
      - whence - `SEEK_SET, SEEK_CUR, SEEK_END, ...`
  - Atualização de registo específico: ler registo, lseek "para trás", escrever novo valor

Permissões



What are the three permission groups?

- **owners** - these permissions will only apply to owners and will not affect other groups.
- **groups**: you can assign a group of users specific permissions, which will only impact users within the group.
- **all users**: these permissions will apply to all users, and as a result, they present the greatest security risk and should be assigned with caution.

What are the three kinds of file permissions in Linux?

- **Read (r)**: Allows a user or group to view a file.
- **Write (w)**: Permits the user to write or modify a file or directory.****
- **Execute (x)**: A user or grup with execute permissions can execute a file or view a directory.

Numeric code

You may need to know how to change permissions in numeric code in Linux, so to do this you use numbers instead of “r”, “w”, or “x”.

- 0 = No Permission
- 1 = Execute
- 2 = Write

- 4 = Read

Permission numbers are:

- 0 = ---
- 1 = --x
- 2 = -w-
- 3 = -wx
- 4 = r-
- 5 = r-x
- 6 = rw-
- 7 = rwx

For example:

- **chmod 777 foldername** will give read, write, and execute permissions for everyone.
- **chmod 700 foldername** will give read, write, and execute permissions for the user only.
- **chmod 327 foldername** will give write and execute (3) permission for the user, w (2) for the group, and read, write, and execute for the users.

> **Note**
>
> `size_t` is an unsigned integer data type that is used to represent the size of an object in bytes. It is defined in the header file `stddef.h` or `stdlib.h`, and it is often used to represent the size of an array, the length of a string, or the size of a memory block.
>
> `ssize_t`, on the other hand, is a signed integer data type that is used to represent the size of a buffer or the number of bytes that have been read or written. It is defined in the header file `unistd.h` and is commonly used in I/O functions to represent the size of data that has been transferred.