

# Sistemas Operativos

2019/2020

# Ficheiros

- Descritor de Ficheiro
  - Representação abstrata de um ficheiro utilizada para operar sobre o mesmo
  - Faz parte da interface POSIX
  - Representado por um inteiro não negativo
  - Pode também servir para representar outros recursos de Input/Output como pipes, sockets, dispositivos de entrada ou saída, e.g. teclado
- Descritores *standard* (podem ser redefinidos - guião 4)

Valor Inteiro	Nome	<unistd.h>	<stdio.h>
0	Standard input	STDIN_FILENO	stdin
1	Standard output	STDOUT_FILENO	stdout
2	Standard error	STDERR_FILENO	stderr

# Estruturas Kernel

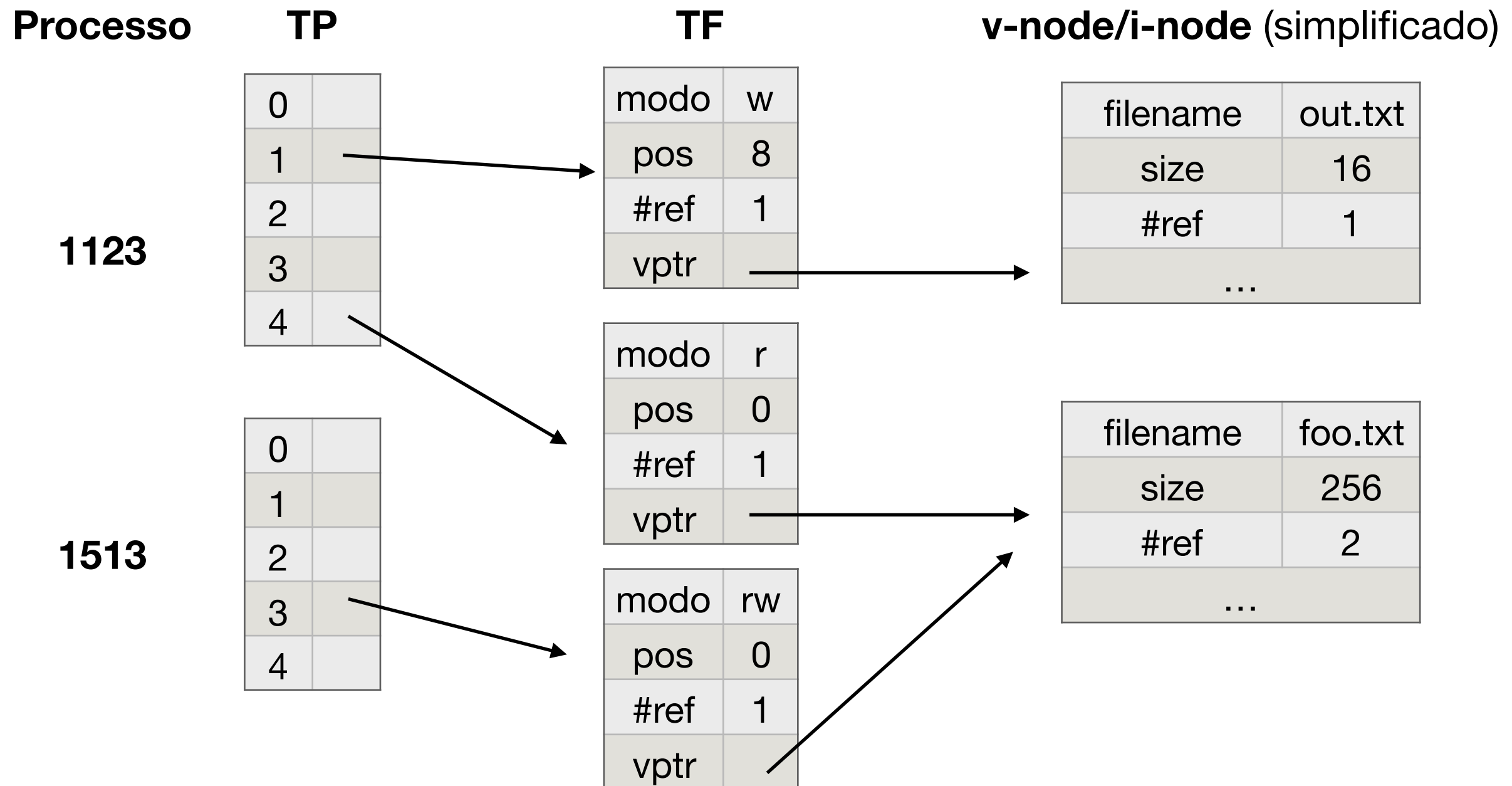
- Tabela de processo (TP)
  - Uma tabela por processo
  - Guarda descritores de ficheiros abertos
  - Use o comando **ulimit -n** para saber quantos ficheiros pode ter abertos
- Tabela de ficheiros (TF)
  - Tabela partilhada pelo sistema operativo
  - Guarda modo de abertura e posição de leitura/escrita de cada descritor

# Estruturas Kernel

- V-node
  - abstração de um objeto Kernel que respeita a interface de 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-node
  - Guarda metadados/atributos do ficheiros (p.ex: nome ficheiro, tamanho, ...)
  - Guarda localização dos dados no recurso físico de armazenamento
- Em Linux, os i-nodes servem também como v-nodes, não havendo uma implementação explícita para os últimos

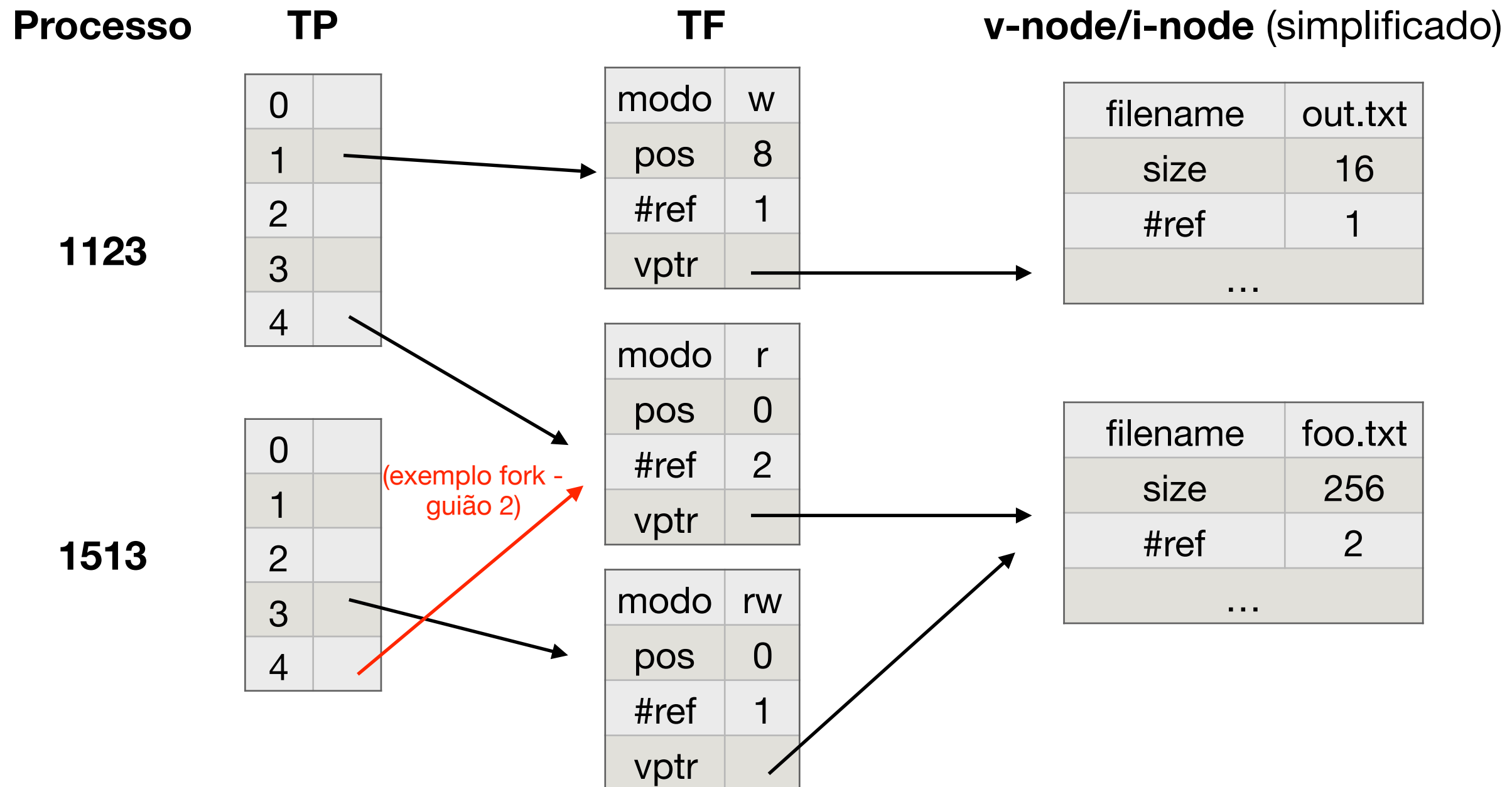
# Estruturas Kernel

- Entradas na tabela de ficheiros de sistema podem partilhar inodes



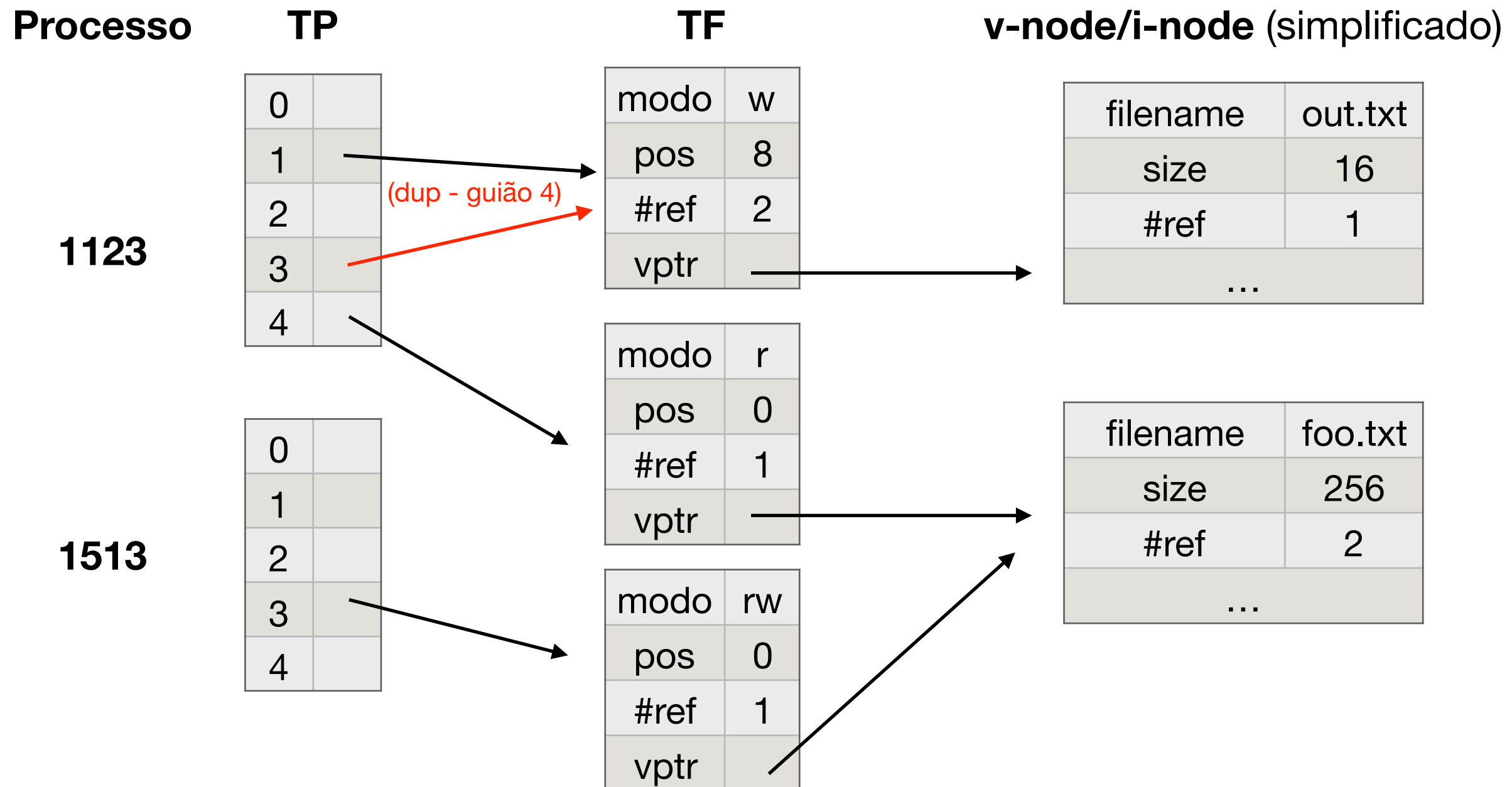
# Estruturas Kernel

- Descritores de **processos distintos (p.ex: via fork)** podem partilhar entradas na tabela de ficheiros de sistema



# Estruturas Kernel

- Descritores do **mesmo processo** (p.ex: **via dup**) podem partilhar entradas na tabela de ficheiros de sistema



# Chamadas ao sistema

- É necessário incluir os cabeçalhos (“headers”):
  - `<unistd.h>` - definições e declarações de chamadas
  - `<fcntl.h>` - definição modos de abertura de ficheiro
    - `O_RDONLY`, `O_WRONLY`, `O_CREAT`, `O_*`



# Chamadas ao sistema

- **int open(const char \*path, int oflag [, mode]);**
  - inicializa um descritor para um determinado ficheiro
  - devolve o descritor ou erro
  - **path** - caminho do ficheiro
  - **oflag** - modo de abertura (O\_RDONLY, O\_WRONLY...)
  - **mode** - permissões de acesso para O\_CREAT (e.g. 0640 equivale a rw-r-----)  
octal ↑

# Chamadas ao sistema

- **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?)*

# Chamadas ao sistema

- **ssize\_t write(int fildes, const void \*buf, size\_t nbyte);**
  - devolve número de bytes escritos ou erro
  - **fildes** - descritor ficheiro
  - **buf** - buffer com conteúdo a escrever
  - **nbyte** - número de bytes a escrever

# Chamadas ao sistema

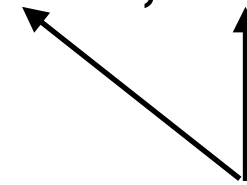
- **int close(int fildes);**
  - apaga o descritor da tabela do processo
  - devolve 0 caso a operação seja executada com sucesso, -1 caso contrário
- **fildes** - descritor ficheiro

# Posição (offset)

- A cada operação de leitura/escrita efetuada sobre o mesmo descritor, a posição a ler/escrever é atualizada consoante o número de bytes efetivamente lidos/escritos

# Exemplo

```
int fd = open("foo.txt", O_CREAT | O_TRUNC | O_RDWR, 0600)
```



**Não esqueça as permissões correctas  
se usar O\_CREAT:      RW => 06 ou 07**

**1513**

0	
1	
2	
3	
4	

# Exemplo

```
int fd = open("foo.txt", O_CREAT | O_TRUNC | O_RDWR, 0600)
```

fd=3

1513

0	
1	
2	
3	
4	

modo	rw
pos	0
#ref	1
vptr	

filename	foo.txt
size	0
#ref	1
...	

# Exemplo

```
ssize_t res = write(fd, "abcde", 5);
```

1513

0	
1	
2	
3	
4	

modo	rw
pos	0
#ref	1
vptr	

filename	foo.txt
size	0
#ref	1
...	



# Exemplo

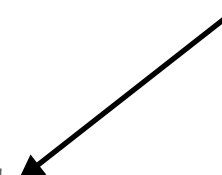
```
ssize_t res = write(fd, "abcde", 5);
```

res=5

**foo.txt**

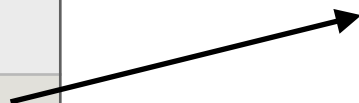
a	b	c	d	e
---	---	---	---	---

posição fd = 5

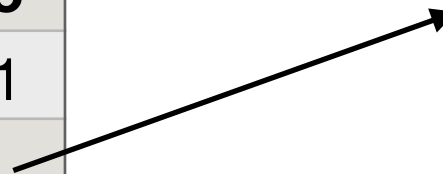


**1513**

0	
1	
2	
3	
4	



modo	rw
pos	<b>5</b>
#ref	1
vptr	



filename	foo.txt
size	5
#ref	1
...	

# Exemplo

```
ssize_t res = write(fd, "fgh", 3);
```

**foo.txt**

a	b	c	d	e
---	---	---	---	---

posição fd = 5

**1513**

0	
1	
2	
3	
4	

modo	rw
pos	<b>5</b>
#ref	1
vptr	

filename	foo.txt
size	5
#ref	1
...	

# Exemplo

```
ssize_t res = write(fd, "fgh", 3);
```

res=3

foo.txt

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

posição fd = 8

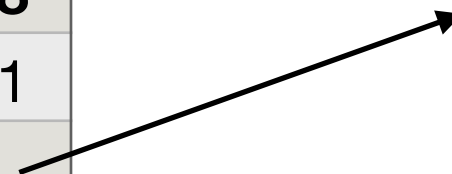


1513

0	
1	
2	
3	
4	



modo	rw
pos	8
#ref	1
vptra	



filename	foo.txt
size	8
#ref	1
...	

# Exemplo

(tentativa de leitura estando posicionado no fim do ficheiro)

```
ssize_t res = read(fd, buf, 8);
```

**foo.txt**

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

posição fd = 8

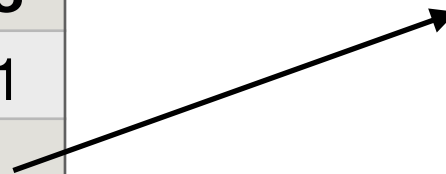


**1513**

0	
1	
2	
3	
4	



modo	rw
pos	8
#ref	1
vptr	



filename	foo.txt
size	8
#ref	1
...	

# Situação de “End of File”

```
ssize_t res = read(fd, buf, 8);
```

**res=0**

**foo.txt**

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

posição fd = 8

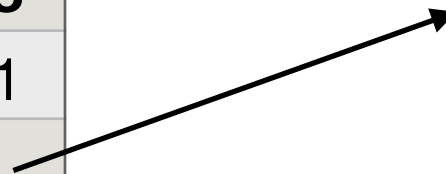


**1513**

0	
1	
2	
3	
4	



modo	rw
pos	8
#ref	1
vp	tr



filename	foo.txt
size	8
#ref	1
...	

# Solução

1. Para acesso sequencial, pode-se criar/ter um outro descritor para leitura do mesmo ficheiro
2. Para uma solução mais geral de acesso directo 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, ...
3. Atualização de registo específico: ler registo, lseek “para trás”, escrever novo valor

# Exemplo

```
int res = close(fd);
```

**foo.txt**

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

posição fd = 8

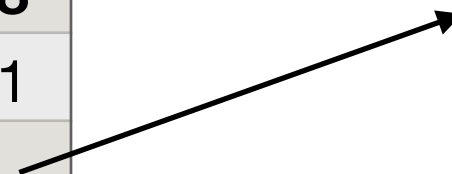


**1513**

0	
1	
2	
3	
4	



modo	rw
pos	8
#ref	1
vp	tr



filename	foo.txt
size	8
#ref	1
...	

# Exemplo

```
int res = close(fd);
```

res=0

**foo.txt**

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

**1513**

0	
1	
2	
3	
4	



# Material Extra

- Outras chamadas
  - `ssize_t pwrite(int fildes, const void *buf, size_t nbyte, off_t offset);`
  - `ssize_t pread(int d, void *buf, size_t nbyte, off_t offset)`
- Leitura:
  - <https://www.usna.edu/Users/cs/aviv/classes/ic221/s16/lec/21/lec.html>
  - <https://www.usna.edu/Users/cs/wcbrown/courses/IC221/classes/L09/Class.html>