

Funções de acesso a directorias

- `DIR *opendir(char *dname);`
- `struct dirent *readdir(DIR *dp);`
 - ler uma entrada (de cada vez)
 - devolve NULL quando atinge o fim
- `int closedir(DIR *dp);`
- `void rewinddir(DIR *dp);`

14-03-2007

ASC II - 06/07

1

Acesso a directorias (2)

- A estrutura DIR é interna à interface e mantém informação sobre a directoria, como uma lista de entradas 'dirent'
- Cada entrada é uma estrutura:

```
struct dirent {
    inot_t d_ino;          /* i-node */
    char d_name[NAME_MAX+1];
}
```

14-03-2007

ASC II - 06/07

2

Exemplo

- Listar os nomes de ficheiros numa directoria

```
dp = opendir("/dir");
if ( dp == NULL ) exit(1);
while ( (dirp=readdir(dp)) != NULL )
    puts(dirp->d_name);
closedir(dp);
```

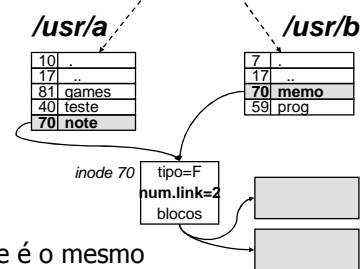
com a chamada `stat()` podia obter os atributos dos ficheiros (informação no inode).

14-03-2007

ASC II - 06/07

3

Múltiplos nomes para um inode?



- `/usr/a/note` é o mesmo ficheiro que `/usr/b/memo`
- cada entrada numa directoria é um *link*

14-03-2007

ASC II - 06/07

4

Definir sinónimos ('hard links')

```
int link(char *antigo, char *novo)
```

- O nome novo fica associado ao mesmo inode que o nome antigo.
- Link cria uma nova entrada numa directoria, com um novo nome, mas com um inode igual ao de um ficheiro existente.
- Um mesmo ficheiro pode ser nomeado por múltiplos nomes diferentes.
- pode-se remover um dos nomes e o ficheiro permanece com o outro nome.

14-03-2007

ASC II - 06/07

5

Múltiplos 'links' de um inode

<code>/usr/a/</code>	<code>/usr/b/</code>
games 81	memo 70
test 40	prog 59

```
link("/usr/b/memo", "/usr/a/note");
```

<code>/usr/a/</code>	<code>/usr/b/</code>
games 81	<u>memo 70</u>
test 40	prog 59
<u>note 70</u>	

14-03-2007

ASC II - 06/07

6

Remover um ficheiro: unlink

- Eliminar da directoria a ligação nome-inode

```
int unlink(char *fname)
```
- Elimina o inode e blocos de dados quando for possível :
 - pode haver outros nomes sinónimos para o mesmo ficheiro (decrementa num. links)
 - pode haver múltiplos canais abertos
 - ficheiro aberto: inode em memória com contador > 0

14-03-2007

ASC II - 06/07

7

Link simbólico (alias)

- Ficheiro especial cujo conteúdo é o nome de outro ficheiro (absoluto ou relativo)
- Definir um link simbólico:
 - comando do *shell*:

```
$ ln -s /usr/lib lib
$ ls -l lib
lrwxrwxrwx 1 user data  lib -> /usr/lib
           nome conteúdo
```
 - chamada ao sistema:

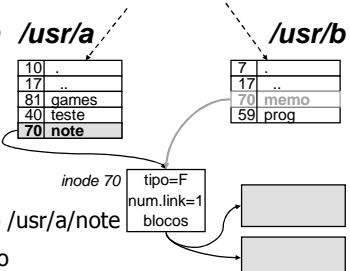
```
int symlink(char *realfile, char *symlink);
```
- `open()` abre o ficheiro real
- `readlink()` permite ler o conteúdo do link

14-03-2007

ASC II - 06/07

8

Renomear ficheiros

- passar `/usr/b/memo` para `/usr/a/note`
- passa a existir como `/usr/a/note`
- deixa de existir como `/usr/b/memo`
- mudou de nome (incluindo a própria directoria)

14-03-2007

ASC II - 06/07

9

Renomear/mover: rename

```
int rename(char *old, char *new)
```

- muda o nome de ficheiros e de directorias
 - permite mudar o nome e mudar de directoria
- garante que o ficheiro não desaparece
- nas directorias mantém a sua consistência (. e ..)

14-03-2007

ASC II - 06/07

10

Criar/remover directorias

- Criar directoria

```
int mkdir(char *dname, mode_t mode);
```

(em vez de creat; e não devolve nenhum descritor)
- Remover directoria

```
int rmdir(char *dname);
```

(em vez de unlink)
 - só remove se a directoria estiver "vazia"

14-03-2007

ASC II - 06/07

11

Directorias (cont.)

- Como escrever numa directoria?
 - só se acrescentar/remover entradas usando as chamadas já vistas (`creat`, `link`, `mkdir`, `unlink`,...)
- São possíveis links para directorias?
 - só se simbólicos
- É possível fazer `unlink` de uma directoria?
 - não (só `rmdir`)

14-03-2007

ASC II - 06/07

12

Generalizando o ficheiro

- O ficheiro pode representar qualquer sequência de bytes
 - em disco, recebida ou enviada pela porta série, em memória, etc...
- O SO oferece a mesma interface ao programador independentemente do dispositivo físico
 - open, read, write, close, ...

14-03-2007

ASC II - 06/07

13

Exemplos: ler ficheiro

```
f=open("/o/ficheiro/a/ler", O_RDONLY);
while( (n=read(f, buf, BUFSIZE))>0 )
    ; /* leu mais um pouco... */
close(f);
```

no fim:

n==0 se chegou ao fim do ficheiro

n==-1 se houve um erro

14-03-2007

ASC II - 06/07

14

Exemplo: ler terminal

```
n = read(0, buf, 1024);
```

- Ler de um terminal é o mesmo que de um ficheiro
 - mas para o SO não é:
 - se o utilizador não escreveu nada tem de esperar que este escreva (não devolve zero)
 - mas não pode esperar que o utilizador escreva todos os caracteres pedidos pelo programa
 - normalmente o SO espera pela introdução de uma linha (quando aparece '\n')

14-03-2007

ASC II - 06/07

15

Dispositivos de tipos diferentes

- Os ficheiros nos discos estão organizados em blocos
 - o SO só escreve/lê blocos de 1 ou mais sectores do disco
- Nos terminais é como se o ficheiro fosse organizado em caracteres
 - pode escrever/ler um número variado de *bytes*

14-03-2007

ASC II - 06/07

16

Exemplos: escrever ficheiro

```
g=creat("/novo/ficheiro", 0666);
n=write(g, buf1, SIZE1);
n=write(g, buf2, SIZE2);
n=write(g, buf3, SIZE3);
close(g);
```

- O SO decide quando realmente envia os dados para o disco
 - p.ex.: periodicamente ou quando enche um novo bloco

14-03-2007

ASC II - 06/07

17

Exemplo: escrever terminal

```
n = write(1, "abcde", 5);
```

- escrever num terminal é o mesmo que num ficheiro
 - mas para o SO não é:
 - cada *write* desencadeia uma verdadeira escrita no periférico

14-03-2007

ASC II - 06/07

18

Máscara de criação de ficheiros

`mode_t umask(mode_t newmask)`

- Cada processo tem um 'umask' associado
- Sempre que cria um ficheiro ou uma directoria, os bits que estiverem a 1 em 'umask' forçam a 0 os bits correspondentes no modo de permissão

- Exemplo: `oldmask = umask(022);`

define: 000 010 010

 - - - -W- -W-

`fd = creat("f", 0666);`

■ em vez ficar: 110 110 110 (0666 = rw-rw-rw-)

■ fica: 110 100 100 (0644 = rw-r--r--)