Sistemi operativi

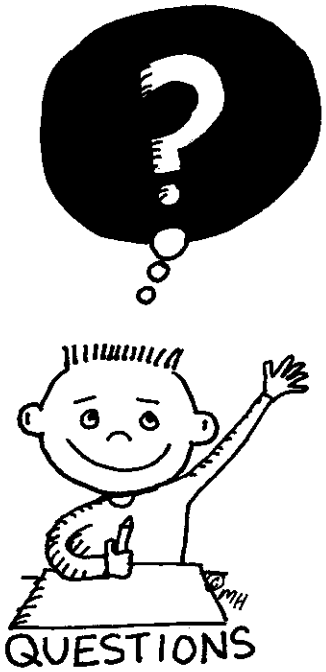
Corso di Laurea Triennale in Ingegneria Informatica

Lezione 10

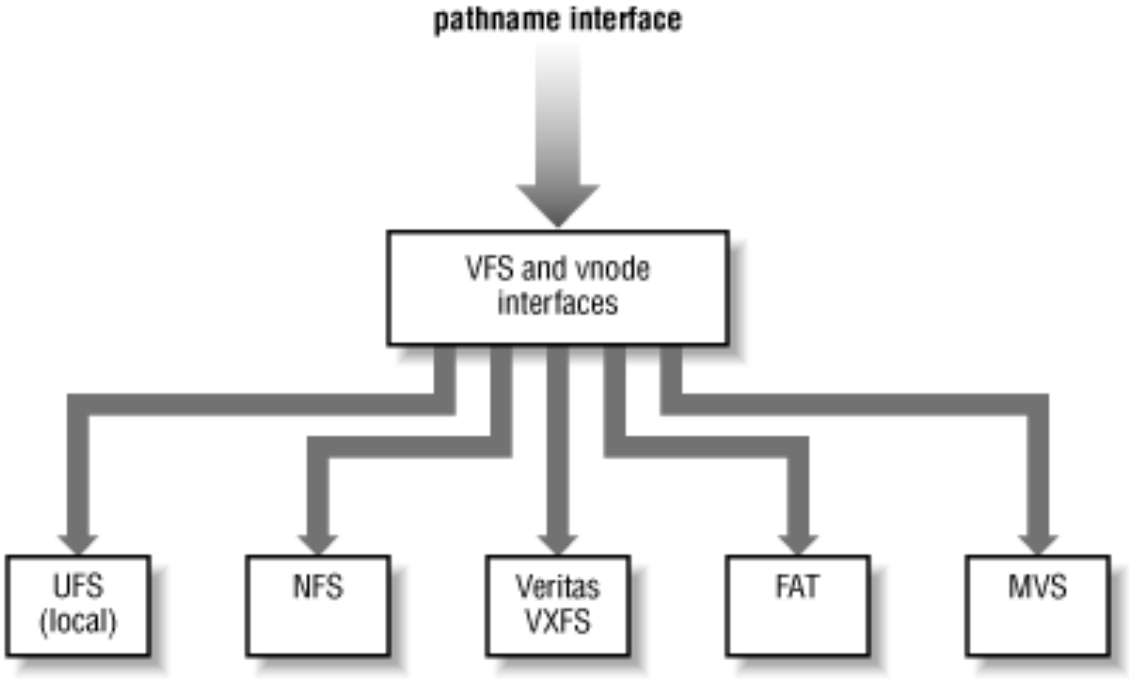
- Virtual Filesystem
- mount, umount
- I/O, Unix I/O,
- Standard I/O
- Pipe e Fifo

Fabio Del Vigna - fabio.delvigna@for.unipi.it

Domande sulle lezioni passate?



Virtual File System



Accesso ai file in UNIX

- **Everything is a file**
  - Garantisce omogeneità del sistema
- Virtual filesystem (**VFS**)
  - interfaccia per l'accesso a filesystem differenti
  - trasparente all'utente.
- Individuazione dei dispositivi
  - periferiche sono riferite come file speciali in `/dev`
  - unità viste come parte di un unico filesystem globale con radice in root (/).

File in Unix (1 / 2)

- Tre categorie di file:
  - File ordinari
  - Direttori (*directory*)
  - Dispositivi (*file speciali*)

- **File: nome, i-node, i-number** (considerare le spiegazioni teoriche)

File in Unix (2 / 2)

- Ad ogni file possono essere associati uno o più nomi simbolici

MA

- Ad ogni file è associato **uno e un solo descrittore (i-node)**, univocamente identificato da un intero (i-number)

Link

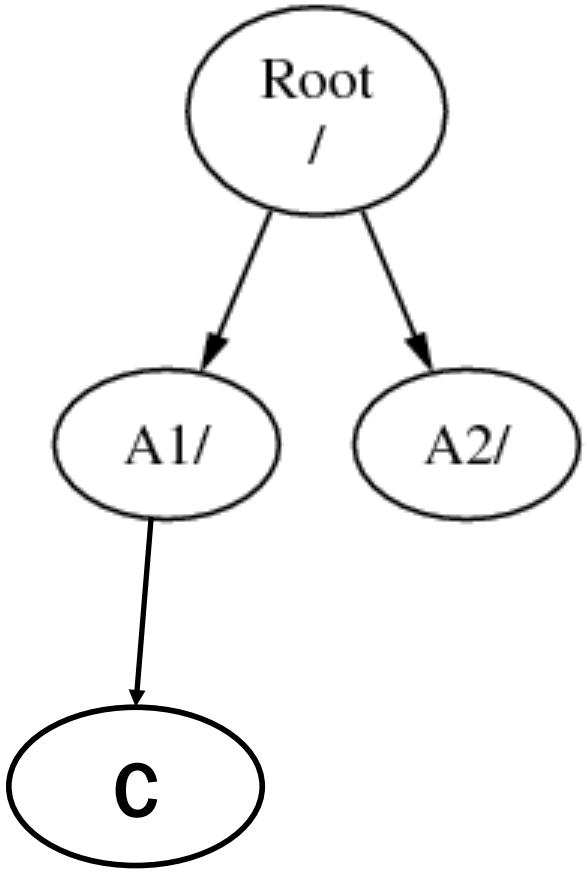
- Si possono creare **link** (collegamenti) a file contenuti nel filesystem
- Vedere l'uso del comando **ln**: `man ln`
- `ln` → Hard link
- `ln -s` → Soft link

i-node

Descrittore del file

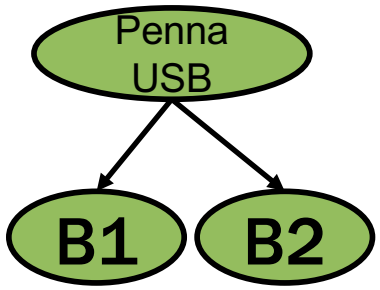
- Tra gli attributi contenuti nell'i-node vi sono:
  1. **tipo di file**: *Ordinario – Direttorio – file speciale*
  2. **proprietario, gruppo** (user-id, group-id)
  3. dimensione
  4. data
  5. **numero di links**

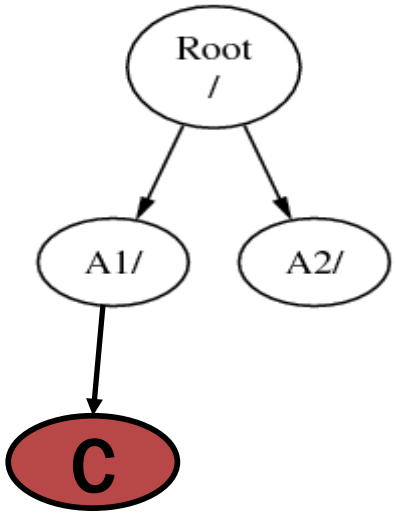
Montare un file system: (1/ 3)



- File system (**F**)
- Supponiamo di avere una penna USB (identificata come file system **B**).
- Per accedere al filesystem **B**, è necessario ***montare*** la penna nel file system (**F**).

Montare un file system: (2 / 3)



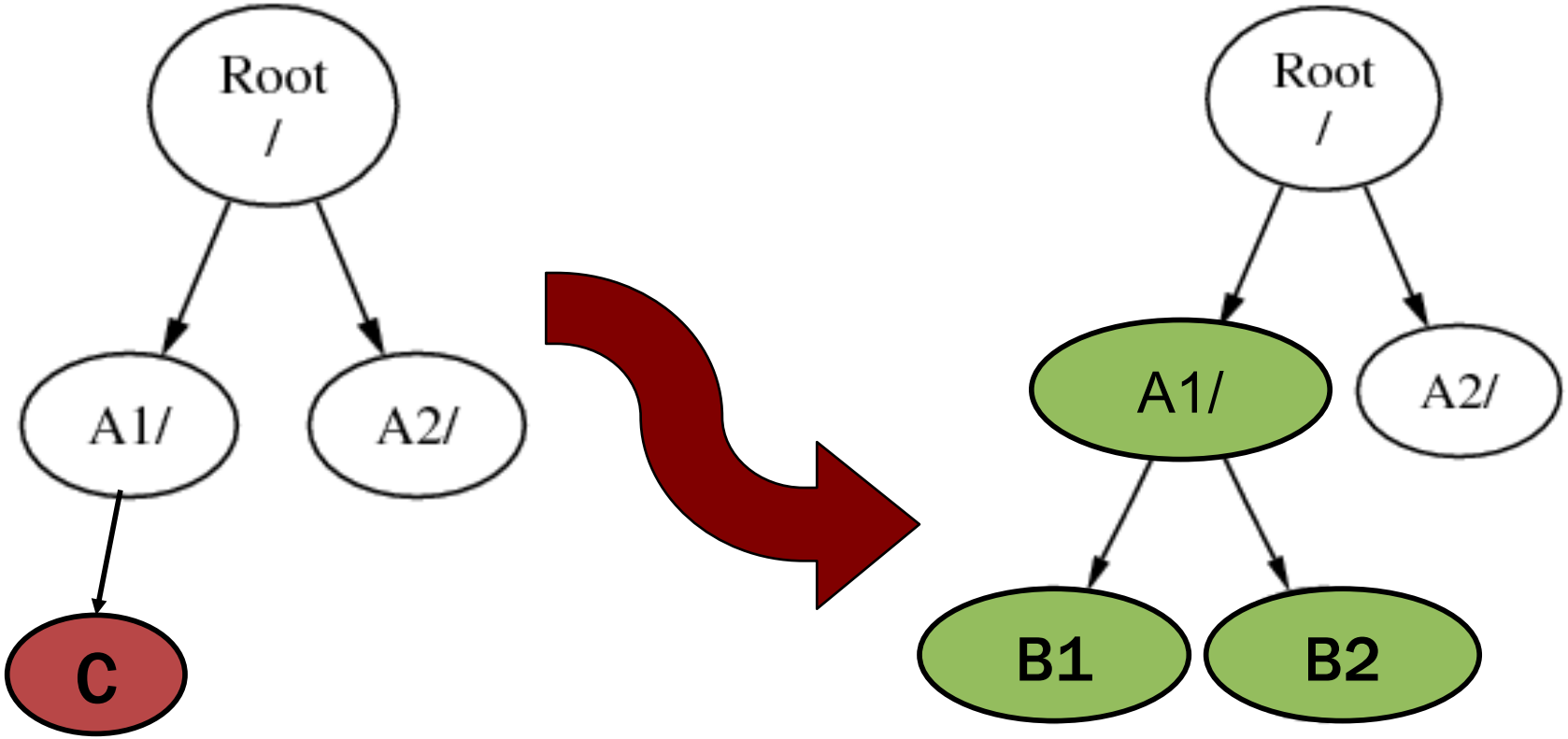


- Scegliere dove “mettere” il file system **B**, es. **A1**
- Montare **B** sulla directory **A1**
- **A1** è “sostituito” dalla directory root del file system **B** contenente le due directory **B1** e **B2**
- I file che erano in **/A1** sono nascosti, tornano visibili quando **B** è smontato da **A1**

Montare un file system: (3 / 3)

Prima di montare il file system B

Dopo aver montato il file system B



Individuazione dei dispositivi in Linux

Dispositivo	Nome
Terminale	`/dev/ttyX`
Hard disk IDE	`/dev/hdX`
Unità ottiche IDE	`/dev/cdromX`
Dischi SCSI e USB mass storage	`/dev/sdX`
Unità ottiche SCSI	`/dev/srX o /dev/scdX`
Lettori floppy	`/dev/fdX`
The black hole	`/dev/null`

Tipi di file system

Tipo di filesystem	Nome
Unix filesystem	`ufs`
Network filesystem	`nfs`
MSDOS (include FAT)	`msdos`
NTFS (Windows NT, 2000, XP, 2003)	`ntfs`
Partizione di scambio	`swap`
ISO9660 (CD-ROM)	`cd9660`
UDF (DVD-ROM)	`udf`

Comando `mount` (1 / 3)

`mount` permette di montare un filesystem su una directory, in modo da rendere accessibili i file e le directory all'interno del file system.

`mount [op.] [filesystem] [mount point]`

- `op.` (opzioni)
  - –t → il tipo del file system da montare
  - -o rw → (lettura/scrittura)
  - -o ro → (solo lettura)
  - …

Comando `mount` (2 / 3)

- **`filesystem`**, dispositivo su cui risiede il file system da montare
- **`mount point`**, directory in cui montare il file system

`mount`, *operazioni preliminari:*

- creare una cartella in cui montare:

```
mkdir /mnt/usb
```

Comando `mount` (3 / 3)

- **Montare un floppy disk**
  - `mount -t msdos /dev/fd0 /mnt/floppy`
- **Montare un CD**
  - `mount -t cd9660 /dev/acd0 /mnt/cdrom`
- **Montare una penna USB**
  - `mount -t msdos /dev/da0 /mnt/usb`

Comando `umount` (1 / 2)

`umount` **smonta i filesytem, cioè esegue l'operazione inversa di** `mount`**.**

`umount [opzioni] [filesystem] [mount point]`

- `filesystem`, **dispositivo su cui risiede il filesystem da smontare**
- `mount point`, **directory da cui smontare il filesystem**

Comando umount (2 / 2)

- `umount`

- Smontare un floppy disk
  - `umount /dev/fd0`
  - `umount /mnt/floppy`
- Smontare un CD
  - `umount /dev/acd0`
  - `umount /mnt/cdrom`
- Smontare una penna USB
  - `umount /dev/da0`
  - `umount /mnt/usb`

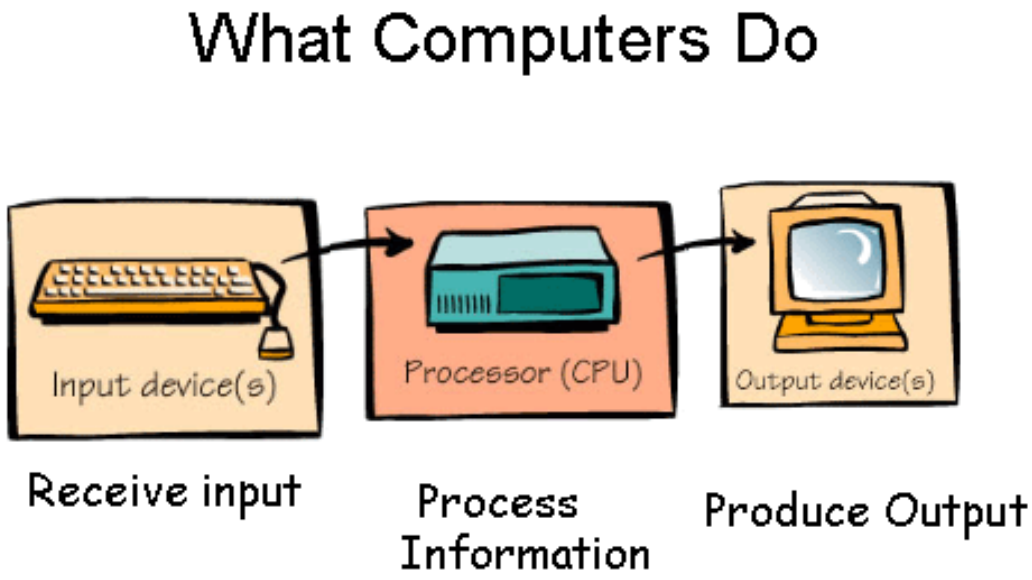
Organizzazione del filesystem

- Il filesystem (a livello logico) di Debian, come quello di altri sistemi Unix, ha un'*organizzazione prefissata*

- Tale organizzazione è detta *filesystem hierarchy*
  - **`man hier`**

I/O

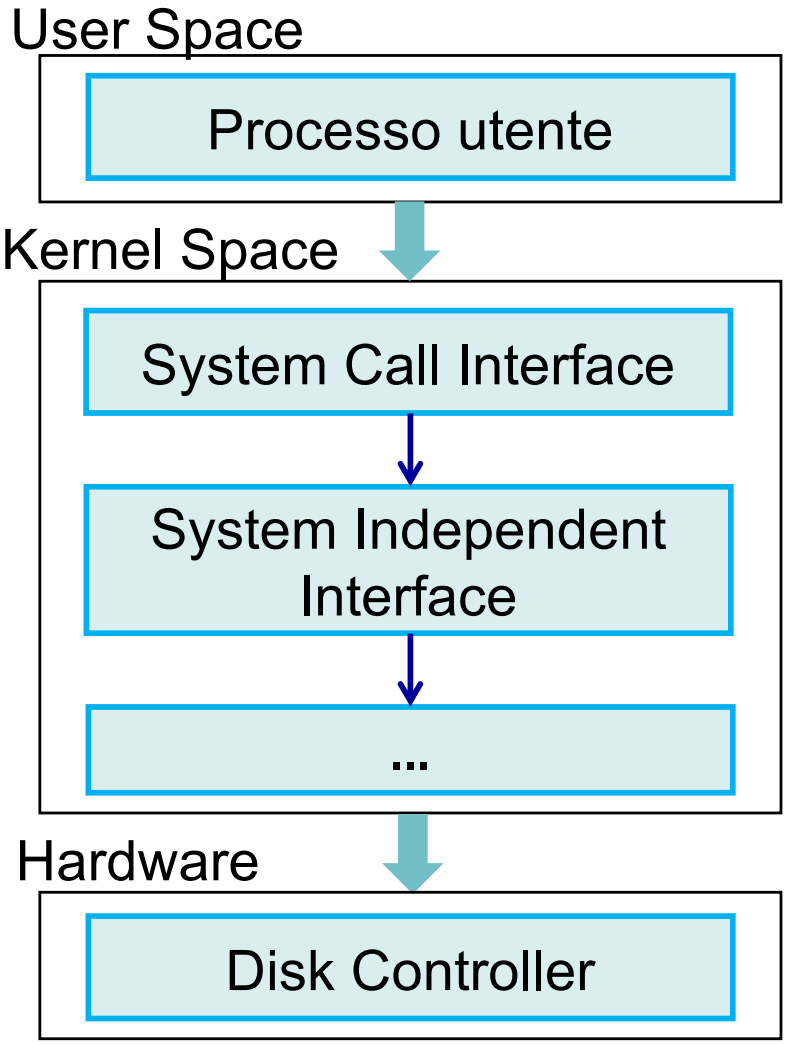
Introduzione



I/O

- L'I/O nei sistemi UNIX è gestito tramite diverse interfacce:
  - UNIX I/O
  - Standard I/O
- Si possono sviluppare altre interfacce a partire da quelle esistenti

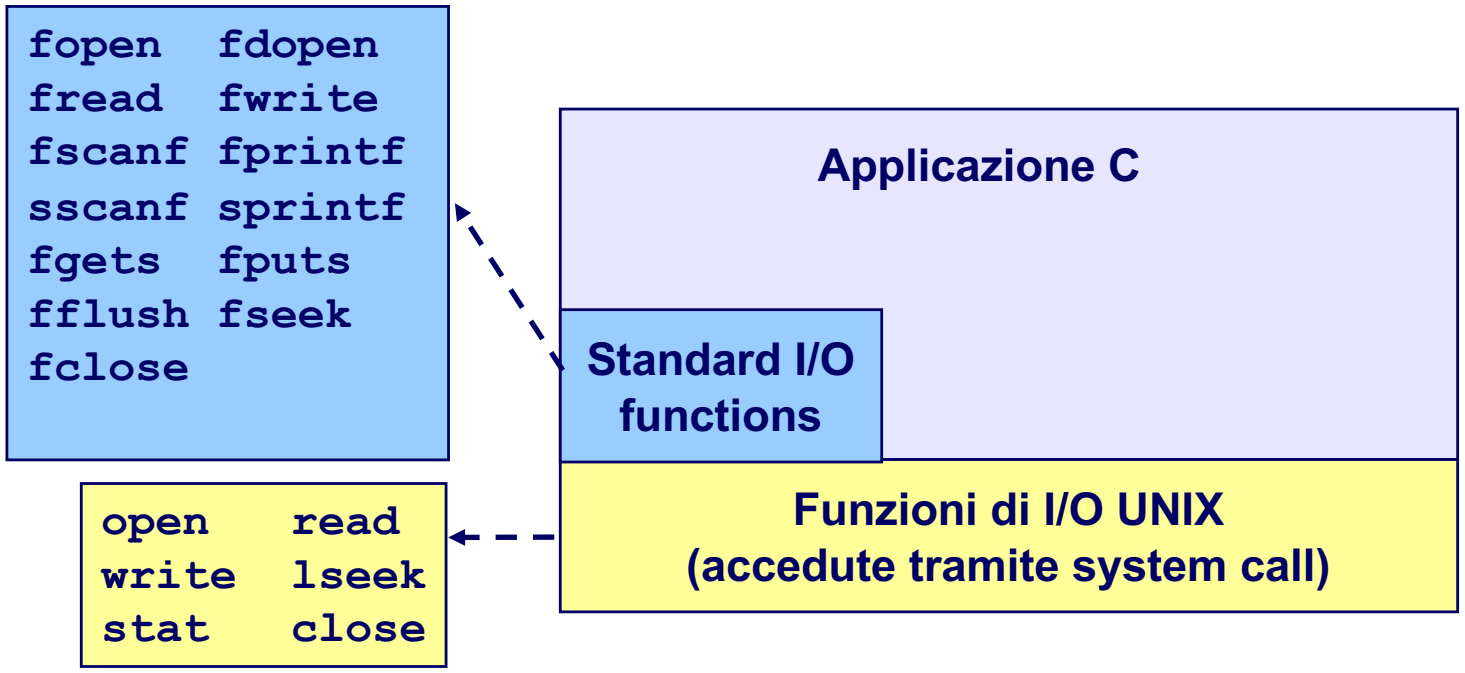
Schema di funzionamento del VFS



- Virtual file system
  - permette la coesistenza di filesystem diversi nello stesso albero delle cartelle,
  - fornisce delle operazioni di input/output **indipendenti** dai dispositivi,
  - i programmi utente possono usare la **stessa interfaccia** per la manipolazione dei file.

Unix I/O vs Standard I/O

Standard I/O è implementato usando l'I/O UNIX (basso livello)

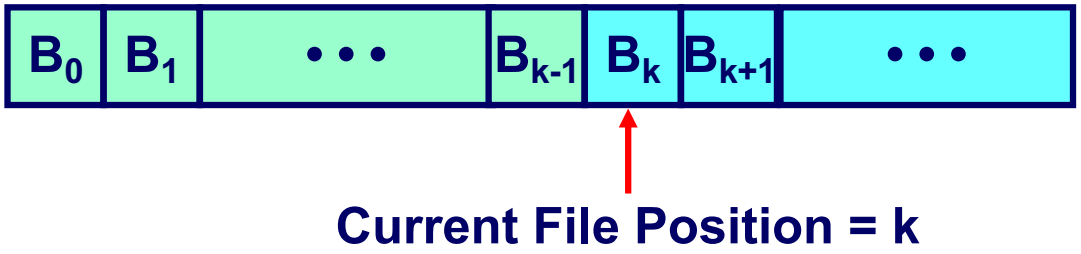


Quale usare?

Unix I/O – System Call

System call di I/O

- Apertura / chiusura di file
  - ⇨ `open()` e `close()`
- Lettura e scrittura di file
  - ⇨ `read()` e `write()`
- Cambiare la *current file position* (seek)
  - ⇨ Indica la posizione da cui leggere o su cui scrivere
  - ⇨ `lseek()`



File aperti di default

Ogni processo creato da una shell Unix inizia la propria esecuzione con tre file aperti associati a un terminale:

- **0: standard input**
- **1: standard output**
- **2: standard error**

Standard I/O - Funzioni

La libreria C standard (`libc.a`) contiene un'insieme di funzioni standard I/O di alto livello

- Aprire / chiudere file (`fopen` and `fclose`)
- Leggere / scrivere byte (`fread` and `fwrite`)
- Leggere / scrivere righe (`fgets` and `fputs`)
- Letture / scritture *formattate* (`fscanf` and `fprintf`)

Riferimenti

The Unix bible:

- Stevens Rago, Advanced Programming in the Unix Environment 2nd edition, Addison Wesley, 2005.

Stevens è probabilmente il miglior autore tecnico di sempre.

- Ha scritto lavori di:
  - Unix programming
  - TCP/IP
  - Unix network programming
  - Unix IPC programming.

Stevens è morto il Sept 1, 1999.

Riferimenti

Parte (quasi tutto) del materiale di queste slide (la parte sull'I/O) è adattamento e traduzione del materiale di http://csapp.cs.cmu.edu/

Pipe

Pipe 1/2

- Le pipe sono degli strumenti di comunicazione tra processi, di tipo monodirezionale
  - Sono spesso utilizzate per “connettere” lo std output di un processo allo std input di un altro
  - Nella Shell sono individuate dall'operatore “|”

Pipe 2/2

- Per il linguaggio C esiste la primitiva

```
int pipe( int fd[2] );
```

- crea una pipe e inserisce l'indice dei suoi descrittori nella struttura fd[]
- Sulla pipe si può operare con i normali operatori per file `write` e `read`.
- Ricordarsi (sempre) di chiudere il “lato” della pipe che non ci serve
  - `close(fd[0])` - input
  - `close(fd[1])` - output

Esempio: pipe e fork (1 di 2)

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
        int     fd[2], nbytes;
        pid_t   childpid;
        char    string[] = "Hello, world!\n";
        char    readbuffer[80];

        pipe(fd);

        if((childpid = fork()) == -1) {
                perror("fork");
                exit(1);
        }
```

Esempio: pipe e fork (2 di 2)

```
        if(childpid == 0) {
                /* Child process closes up input side of pipe */
                close(fd[0]);
                /* Send "string" through the output side of pipe */
                write(fd[1], string, (strlen(string)+1));
                exit(0);
        }
        else {
                /* Parent process closes up output side of pipe */
                close(fd[1]);
                /* Read in a string from the pipe */
                nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
                printf("Received string: %s", readbuffer);
        }
        return(0);
}
```

Altre funzioni sulle pipe

```
FILE *popen( char *command, char *type)

int pclose( FILE *stream )
```

file: **popen1.c**

```
popen("ls ~scottb", "r");

popen("sort > /tmp/foo", "w");
```

file: **popen2.c**

file: **popen3.c**

```
popen3 sort popen3.c

popen3 cat popen3.c

popen3 cat popen3.c | grep main
```

FIFO

FIFO

- Sono simili alle pipe (named pipe).
- Non sono gestite nel kernel, ma nel filesystem
- Hanno associato un nome (a differenza delle pipe)

- Creare una FIFO

```
mknod MYFIFO p
mkfifo a=rw MYFIFO
```

```
int mknod( chat *pathname, mode_t mode, dev_t
  dev);
```

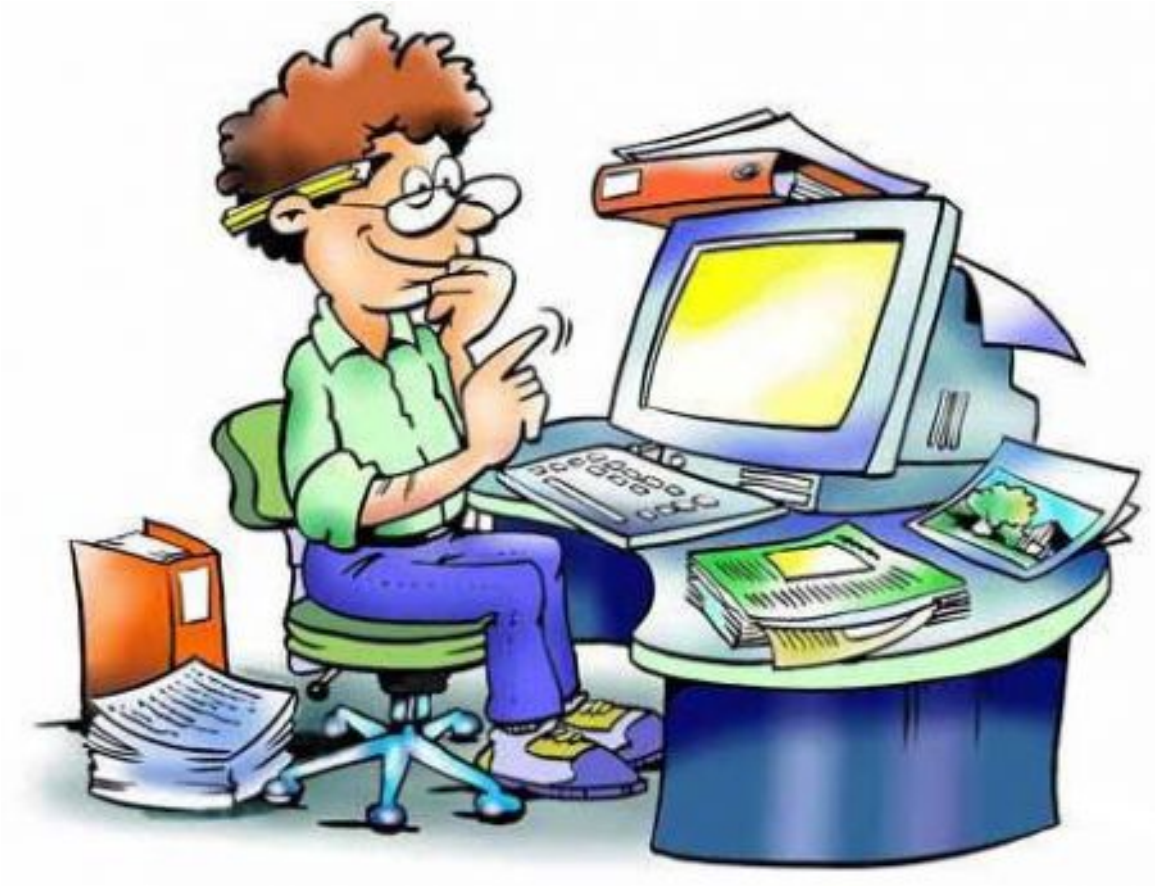
Lavorare sulle FIFO

- Una volta creata la FIFO possiamo lavorarci come su un file
- Hanno come le pipe un “senso” di utilizzo
- Vanno “aperte” con una open e chiuse con una close
- All'atto dell'apertura specificare se “r” o “w”

- `file: fifoserver.c fifoclient.c`

```
./fifoserver&
./fifoclient
```

Esercizi



Esercizi (1 / 2)

1. Eseguire il login come utente **root.**
2. Creare un utente **pippo** utilizzando il comando `adduser -s` (**con home la cartella** `/tmp/pippo`).
3. Creare un nuovo gruppo **usbkeyusr** a cui deve appartenere l'utente **pippo.**
4. Creare una directory `/tmp/usbkey` e montare la penna usb al suo interno. (bisogna prima sapere il nome che il sistema ha assegnato al dispositivo usb ...)
5. Provare con l'utente **pippo** a modificare il contenuto della penna usb.
6. Impostare **usbkeyusr** come group owner della directory `/tmp/usbkey` e assegnare alla cartella i seguenti diritti: accesso illimitato per i membri del gruppo, nessun tipo di accesso per gli altri.

Esercizi (2 / 2)

7. Verificare che l'utente **pippo** riesca a creare, visualizzare, cancellare file e cartelle nella penna usb.
8. Smontare la penna usb.
9. Eliminare l'utente **pippo.**

Esercizio 2

1. **Eseguire il login come utente root.**
2. **Giocate con** `ln, mtools, mdconfig, dd, ... altri comandi utili per lavorare sui filesystem ...`
3. **Cosa avete scoperto?**
4. **Che dubbi avete?**

Soluzione (1 / 2)

Login root

```
adduser -s [...]
```

```
vi /etc/group
```
: aggiungere la riga (es.):
```
    usbkeyusr:*:1003:pippo
```
Attenzione: scegliere come ID di gruppo un valore non in uso

```
mkdir /tmp/usbkey
    mount -t msdos /dev/DEVNAME /tmp/usbkey
```

Non è permesso.

```
umount /tmp/usbkey
```
(altrimenti i comandi seguenti non hanno effetto)
```
    chown :usbkeyusr /tmp/usbkey
```

Soluzione (2 / 2)

```
   chmod g+rwx,o-rwx /tmp/usbkey
   mount -t msdos /dev/DEVNAME
   /tmp/usbkey
…
umount /tmp/usbkey
rmuser pippo
```

Soluzione esercizio 2

Non c'è soluzione ...

..., giocate, sperimentate, imparate