UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

SISTEMI OPERATIVI (080IN)

A.A. 2021 / 2022

Lezioni 2 - Interfacce di Sistemi Operativi

Sylvio Barbon Junior
sylvio.barbonjunior@units.it

Sommario:

1) Lezione precedente;
2) Cos'è un Sistema Operativo;
3) Processo e memoria
4) I/O e File Descriptor
5) Pipes
6) File System

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

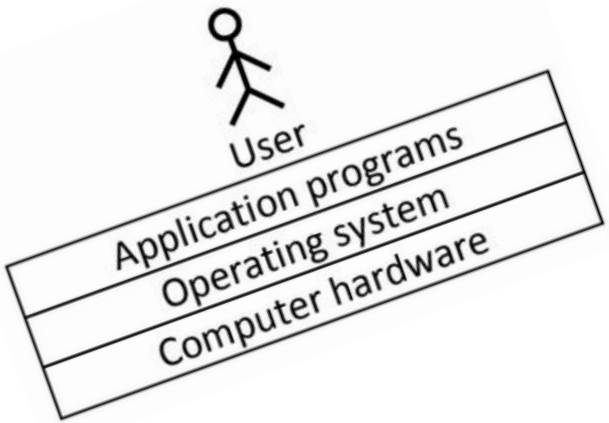
2) Motivazioni principali del corso

- Il corso **NON** si occupa di utilizzo dei Sistemi Operativi a livello utente;
- Di familiarizzare con Concetto dei Sistemi Operativi (Windows, Linux, MacOS, Android ecc);
- Di conoscenza approfondita del linguaggio C;
- Di familiarizzare con la programmazione dei calcolatori.

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) Lezione precedente

- Hardware: è la collezione di sistemi fisiche che abbiamo nel calcolatore.
- Software: è la collezione di tutti i programmi memorizzati ed eseguiti per il calcolatore;



Uno Sistema Operativo deve:

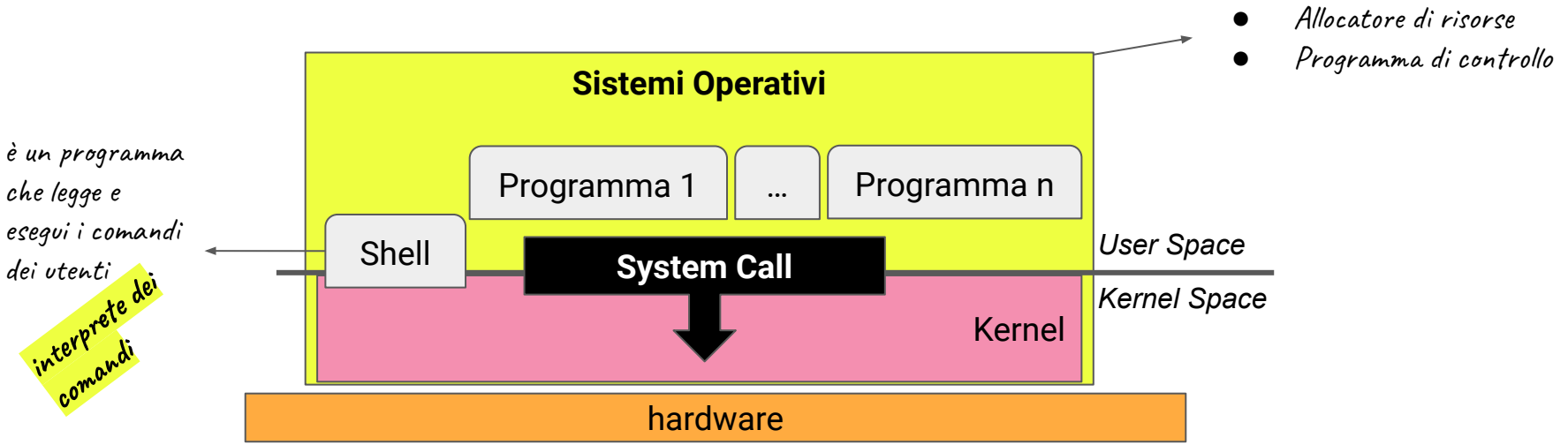
- gerire l'hardware del calcolatore;
- facilitare gli esecuzioni dei programmi applicativi;
- essere efficiente;
- essere conveniente;

Personal Computer	Facile di usare
High Performance C.	uso adeguato delle risorse
Dispositivi palmari	risparmio energetico
Dispositivi embedded	interface minima con utente

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

2) Cos'è un Sistema Operativo

- Sistema operativo: insieme di programmi (software) che gestisce gli elementi fisici di un calcolatore (hardware);
- SO basato su Unix assume la forma tradizionale di un kernel (nucleo), che è un programma speciale che fornisce servizi ai programmi in esecuzione;

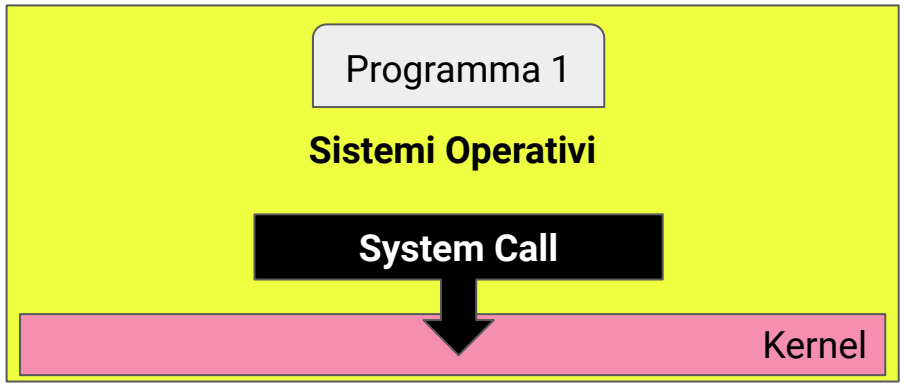


2) Cos'è un Sistema Operativo

una raccolta di codice in formato testo, scritta utilizzando un linguaggio di programmazione comprensibile dall'uomo

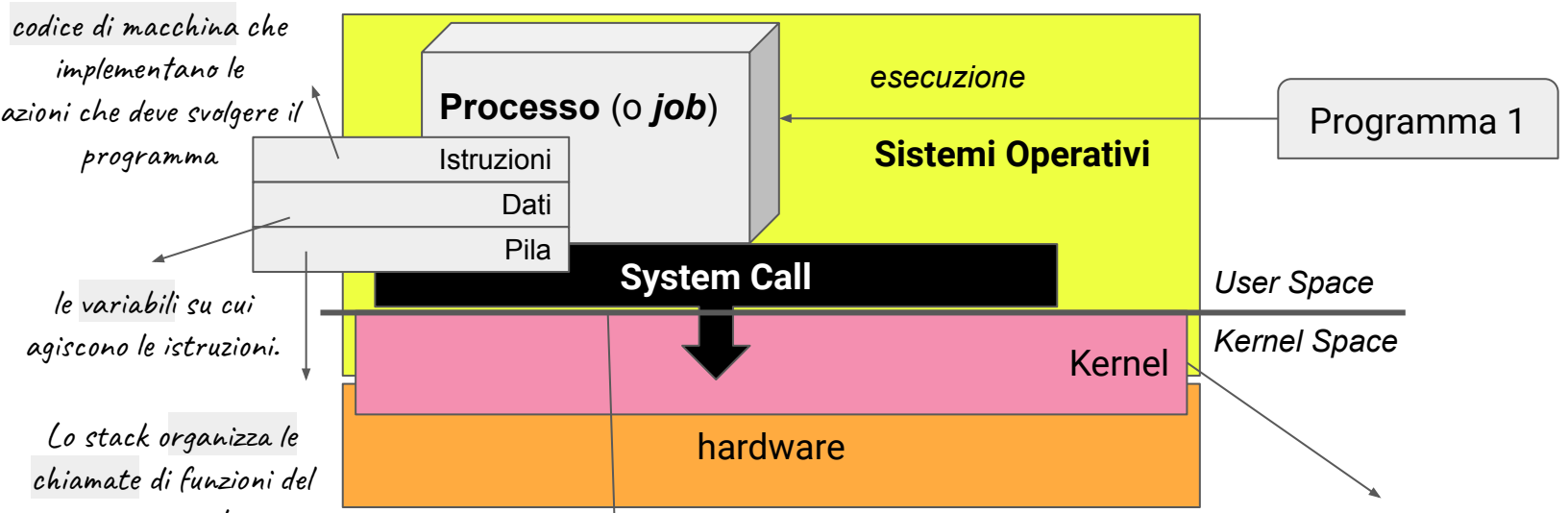
Il codice sorgente viene trasformato da un assemblatore o compilatore in codice macchina che può essere eseguito dal computer.

Il codice macchina viene memorizzato dal compilatore in un **file** (programma) per l'esecuzione in un secondo momento.



il codice sorgente può essere interpretato e quindi eseguito immediatamente.

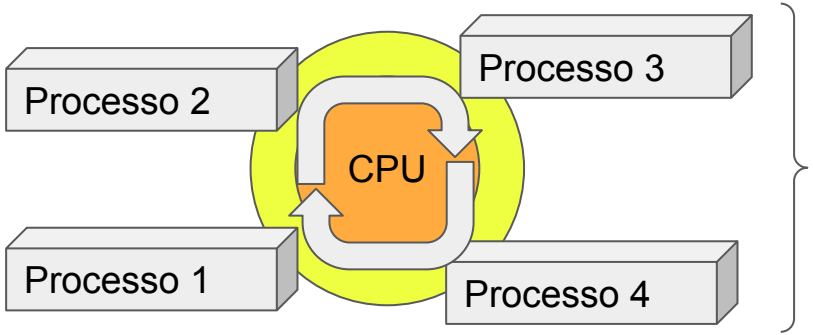
2) Cos'è un Sistema Operativo



Kernel è quella parte del sistema operativo che interagisce direttamente con l'hardware e svolge i compiti più cruciali.

3) Processo e Memoria

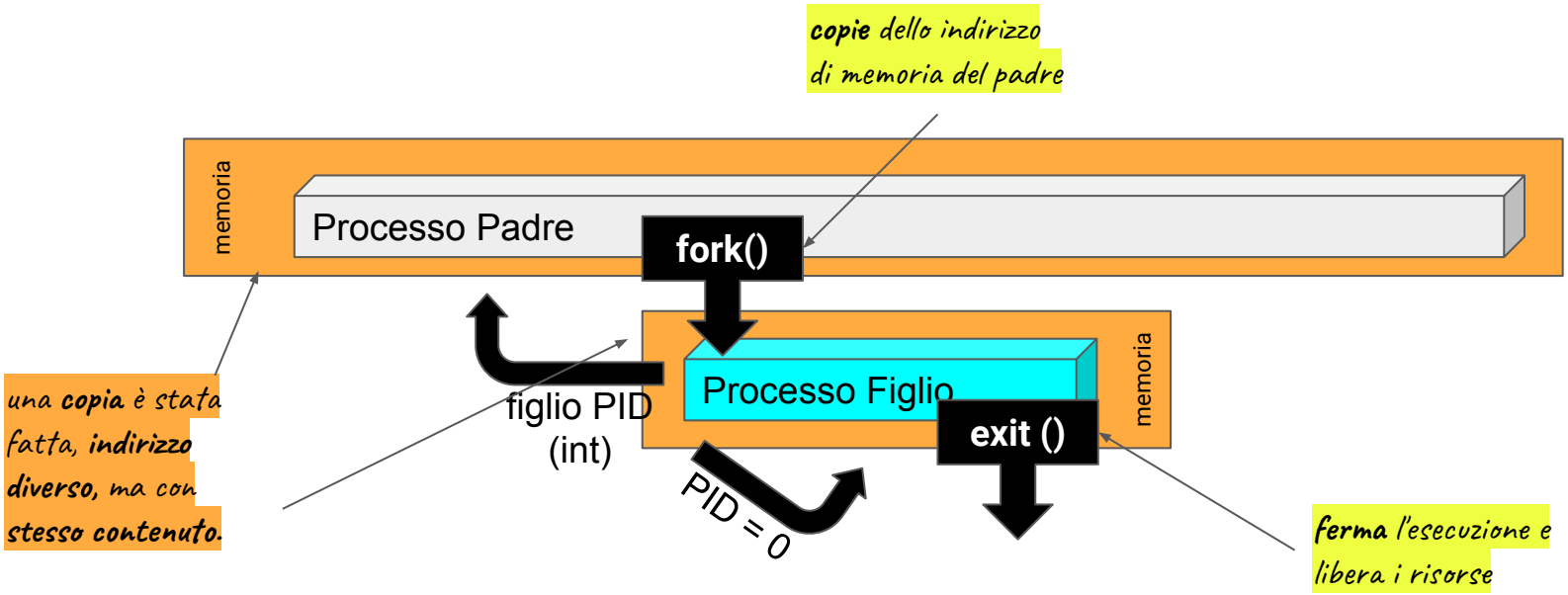
- Un processo nel spazio utente è costituito da memoria (istruzioni, dati e stack) e stato privato al kernel.
- L'idea di processi time-share riguarda, di modo trasparente, la commuta del CPU disponibili tra l'insieme di processi in attesa di esecuzione.
- Quando un processo non è in esecuzione, SO salva i suoi registri della CPU, ripristinando li alla successiva esecuzione il processo.
- Il kernel associa un identificatore di processo, o pid, a ogni processo.
- The kernel associates a process identifier, or **pid**, with each process




```
root@barbon-XPS13-9333: /home/barbon
top - 17:13:51 up  1:58,  1 user,  load average: 0,78, 0,97, 1,39
Tasks: 316 total,   1 running, 315 sleeping,   0 stopped,   0 zombie
%Cpu(s): 14,2 us,  4,8 sy,  0,0 ni, 80,9 id,  0,0 wa,  0,0 hi,  0,1 si,  0,0 st
MiB Mem :   7855,9 total,    151,2 free,   5573,2 used,   2131,5 buff/cache
MiB Swap:   2048,0 total,    611,5 free,   1436,5 used.    892,3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1490 barbon    20   0  970744 113288  58260 S  16,9   1,4   8:17.47 Xorg
  13267 barbon    20   0  818860  44648  33672 S  11,3   0,6   0:01.28 gnome-s+
   1663 barbon    20   0 4294772 142124  46148 S   9,9   1,8   7:34.22 gnome-s+
  11851 barbon    20   0 9114348 279732 107208 S   9,3   3,5   1:40.79 Isolate+
   1427 barbon     9 -11 5162044  17840  14252 S   7,3   0,2   6:42.57 pulseau+
  12033 barbon    20   0 2994840 234720 136924 S   6,6   2,9   1:09.81 spotify
   2066 barbon    20   0   38,8g 101780  25364 S   2,6   1,3   2:17.46 skypefo+
   3458 barbon    20   0 4456420 486876 180820 S   2,3   6,1  22:35.35 GeckoMa+
   7668 barbon    20   0   13,5g 350160  52432 S   2,3   4,4  14:34.86 teams
   2040 barbon    20   0 1083664  18464  14648 S   2,0   0,2   1:42.24 skypefo+
   4538 barbon    20   0   25,1g 129316  78784 S   1,3   1,6   0:22.18 chrome
```

3) Processo e Memoria

- fork trasforma un singolo processo in due processi identici, riconoscibili come processo padre e processo figlio.



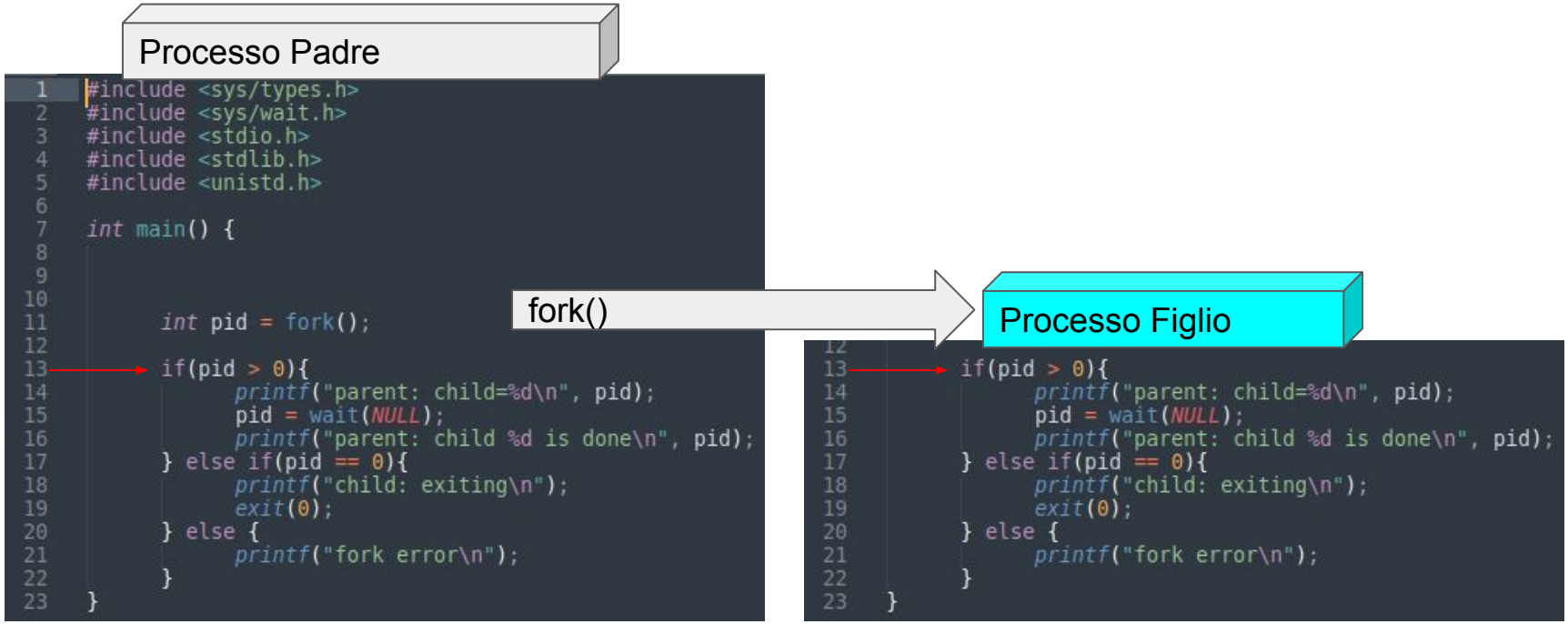
3) Processo e Memoria

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

int main() {



        int pid = fork();

        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

```
barbon@barbon-XPS13-9333: ~/Downloads/SO/CodiceSorge...
(base) barbon@barbon-XPS13-9333:~/Downloads/SO/CodiceSorgente$ gcc -o processo1
 processo1.c
(base) barbon@barbon-XPS13-9333:~/Downloads/SO/CodiceSorgente$ ./processo1
parent: child=23335
child: exiting
parent: child 23335 is done
(base) barbon@barbon-XPS13-9333:~/Downloads/SO/CodiceSorgente$
```

int wait() forza un processo padre ad aspettare che un processo figlio si fermi oppure termini.

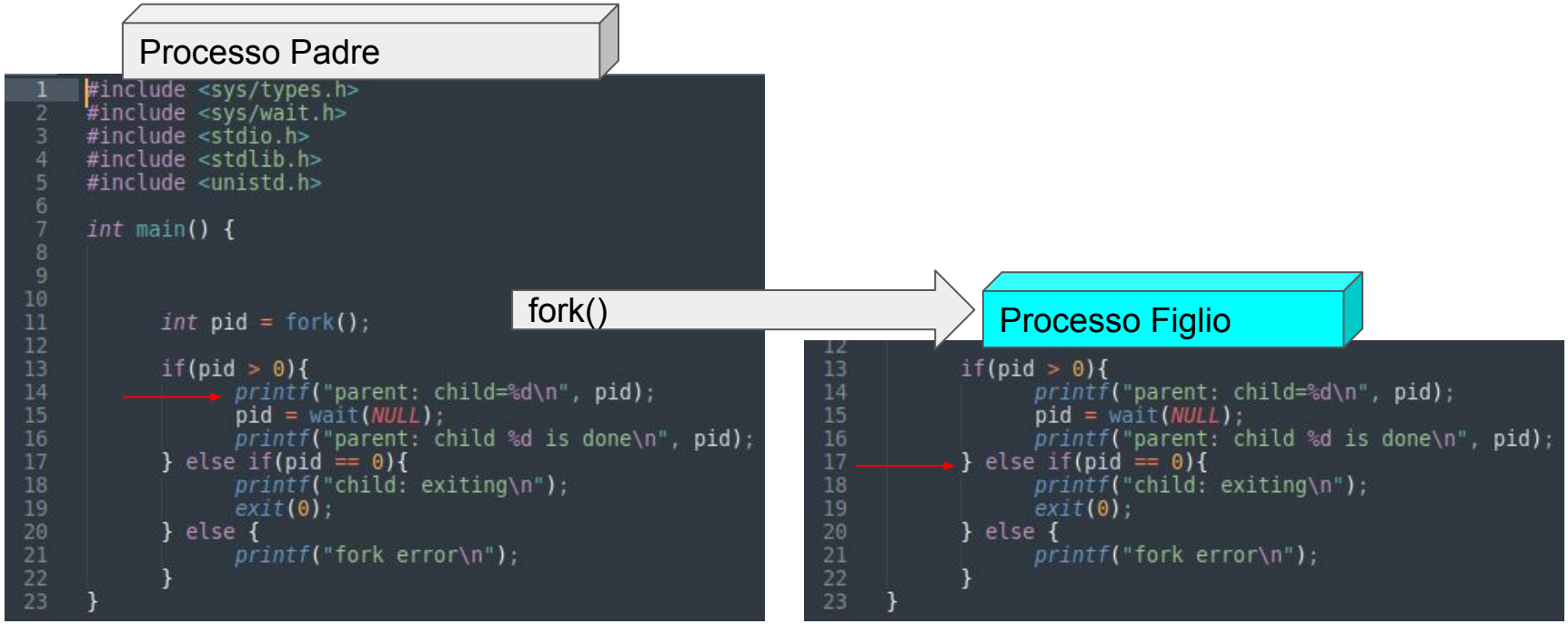
3) Processo e Memoria




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

int main() {


        int pid = fork();

        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

```c
        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

3) Processo e Memoria




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

int main() {


        int pid = fork();

        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

```c
        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

3) Processo e Memoria

Processo Padre

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

int main() {



        int pid = fork();

        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

fork()

Processo Figlio

```c
        if(pid > 0){
                printf("parent: child=%d\n", pid);
                pid = wait(NULL);
                printf("parent: child %d is done\n", pid);
        } else if(pid == 0){
                printf("child: exiting\n");
                exit(0);
        } else {
                printf("fork error\n");
        }
}
```

3) Processo e Memoria

Processo Padre

```
1  #include <sys/types.h>
2  #include <sys/wait.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6
7  int main() {
8
9
10
11        int pid = fork();
12
13        if(pid > 0){
14              printf("parent: child=%d\n", pid);
15              pid = wait(NULL);
16              printf("parent: child %d is done\n", pid);
17        } else if(pid == 0){
18              printf("child: exiting\n");
19              exit(0);
20        } else {
21              printf("fork error\n");
22        }
23 }
```

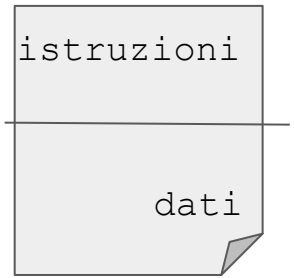
Processo Figlio

```
13        if(pid > 0){
14              printf("parent: child=%d\n", pid);
15              pid = wait(NULL);
16              printf("parent: child %d is done\n", pid);
17        } else if(pid == 0){
18              printf("child: exiting\n");
19              exit(0);
20        } else {
21              printf("fork error\n");
22        }
23 }
```

3) Processo e Memoria

"file path" punta al nome di un file contenente un comando che deve essere eseguito, a

"arg1, ..., argn" sono puntatori agli argomenti per il comando



exec(file path, *arg1)

1

Il spazio de **memoria** dello utente (user-space) è a assegnata implicitamente.

fork()

assegna come una **copia** della memoria **del padre**

exec(file path, *arg1)

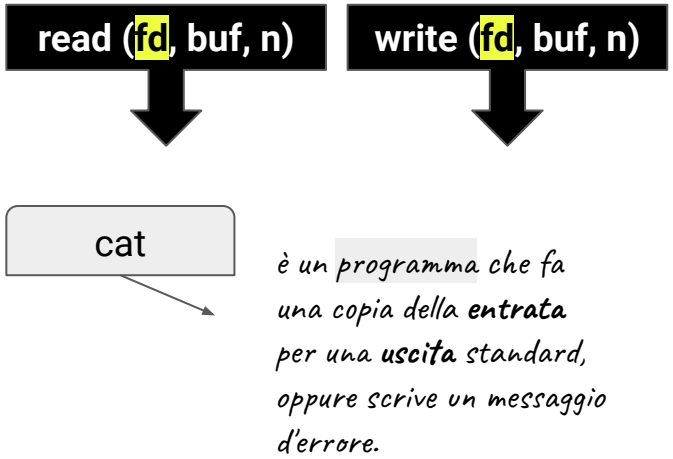
assegna la memoria sufficiente **secondo il file**

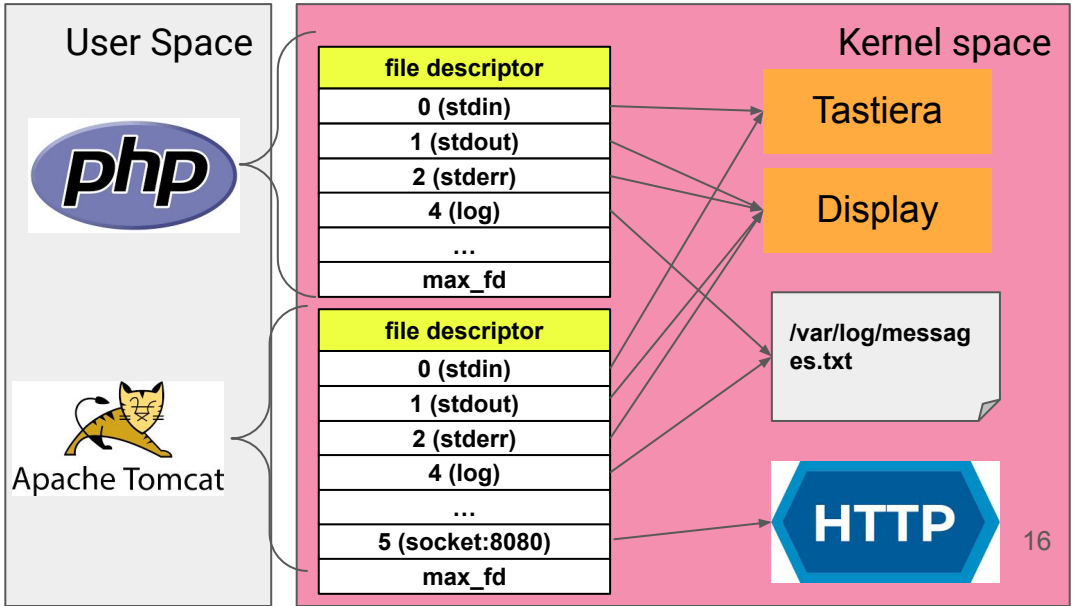
2

Se un processo bisogna di piu memoria in run-time, deve assegnare piu.

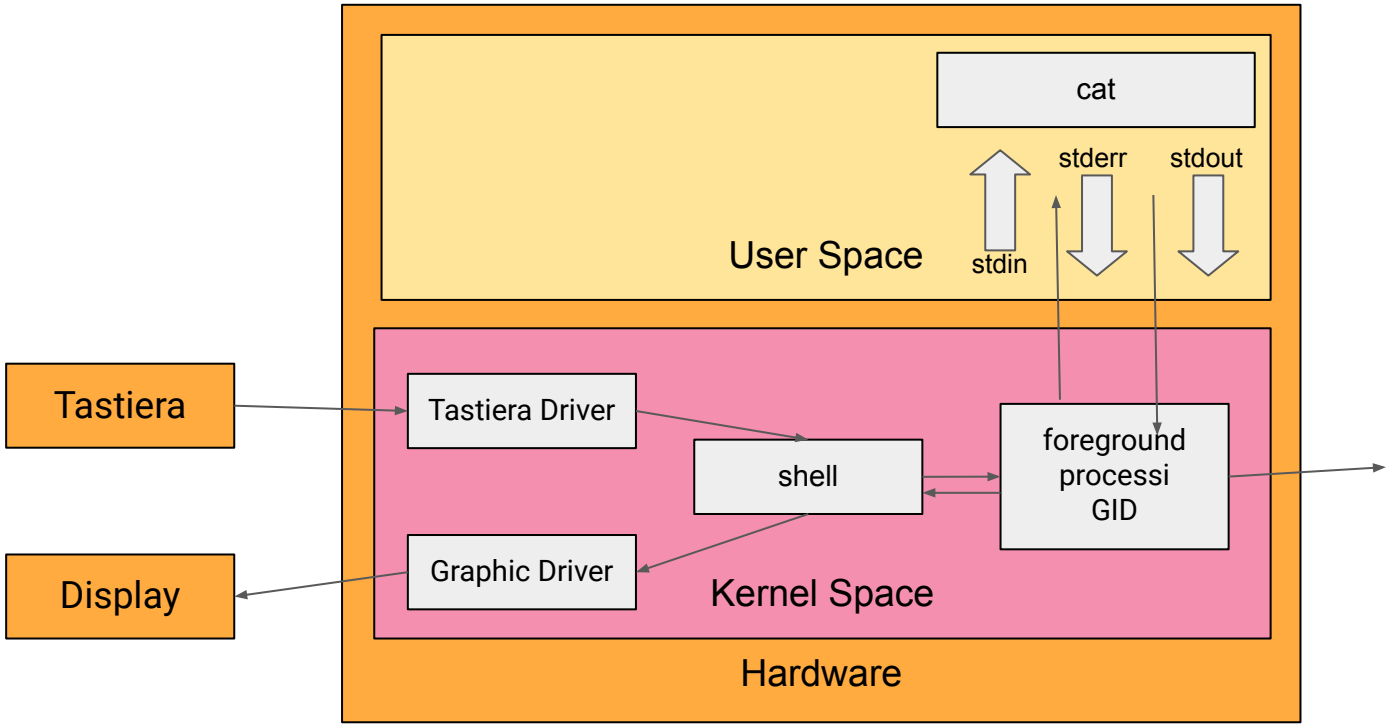
4) I/O e Files

- **file descriptor (descrittore di file): fd,** è un numero intero che rappresenta un file, directory, pipe o socket. La ideia nell'ambito della **memoria RAM**.
- Ogni processo ha la sull **fd**
- La Systems Calls che lavorano con Files e I/O dipende dello **fd**.





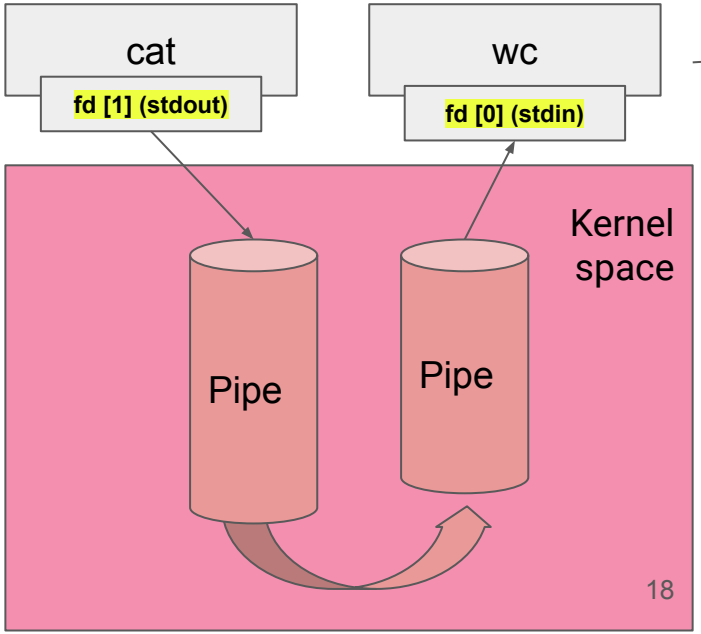
4) I/O e Files



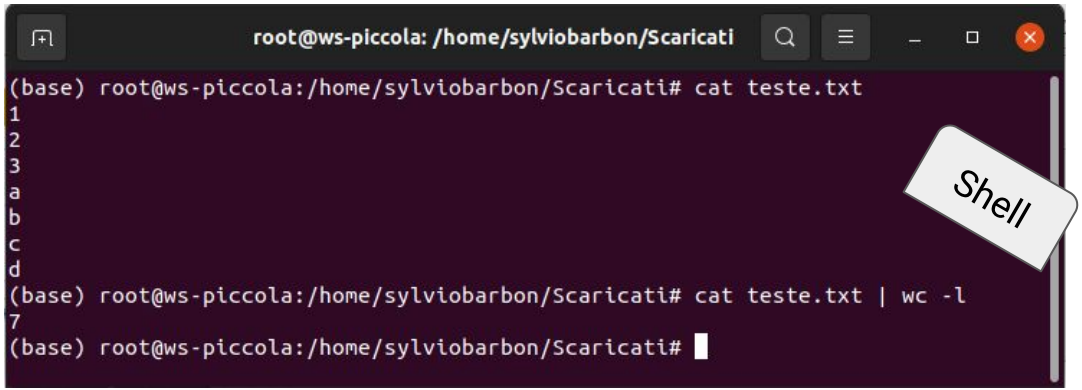
foreground process group identification (GID) sono quello eseguiti in primo piano.

5) Pipes

- La pipe è un canale di comunicazione tra processi, basato in una coppia di fd, un per reading e altro per writing.



wc è un programma che conta la quantità di parole, righe ed ecc



6) File System

- Dal punto di vista dell'utente, un file system è composto da due elementi:
  - **file**: unità logica di memorizzazione
  - **directory**: un insieme di informazioni per organizzare e fornire informazioni sui file che compongono un file system

Il concetto di file:

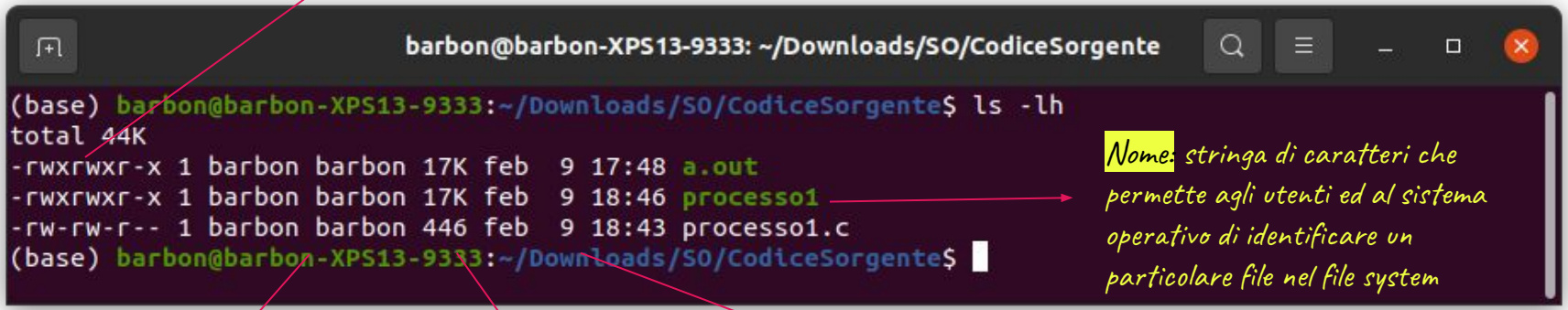
- è l'entità atomica di assegnazione/gestione della memoria secondaria
- è una collezione di informazioni correlate
- fornisce una vista logica uniforme ad informazioni correlate

Tipi di file:

- A seconda della struttura interna:
  - senza formato (stringa di byte): file testo, ecc
  - con formato: file di immagine, file di database, a.out,...
- A seconda del contenuto
  - ASCII/binario (visualizzabile o no, 7/8 bit)
  - sorgente, oggetto, ecc
  - eseguibile (oggetto attivo)

6) File System

Attributi di protezione: informazioni di accesso per verificare chi è autorizzato a eseguire operazioni sui file



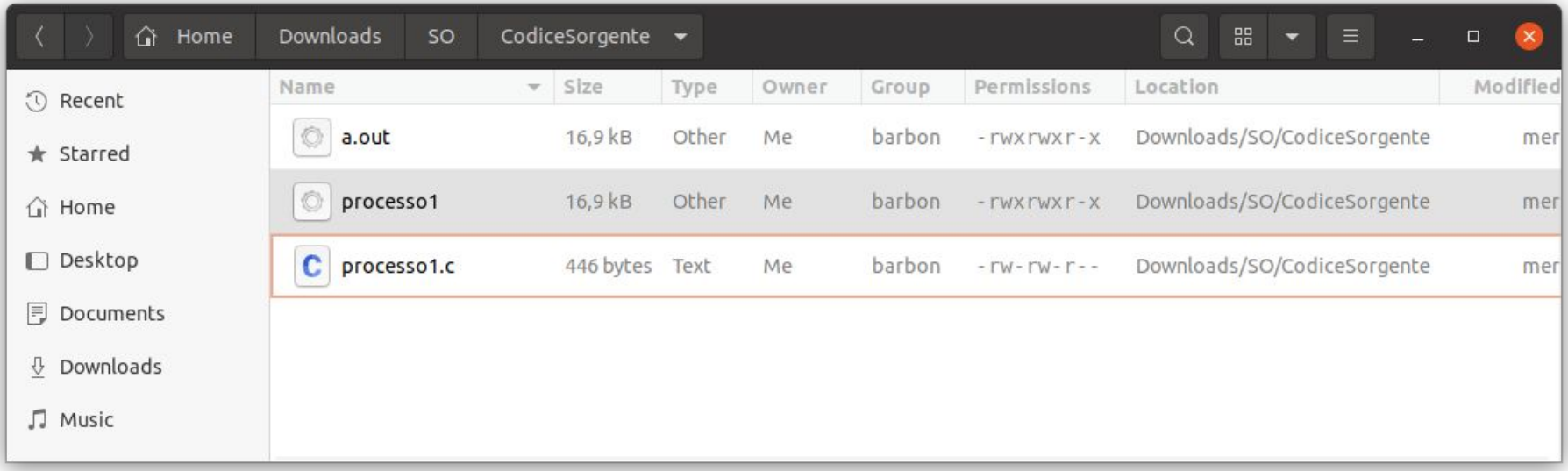
Nome: stringa di caratteri che permette agli utenti ed al sistema operativo di identificare un particolare file nel file system

Informazioni sulla proprietà: utenti, gruppi, ecc, è utilizzato per accounting e autorizzazione

Dimensione: informazioni sul costo del file in memoria secondaria

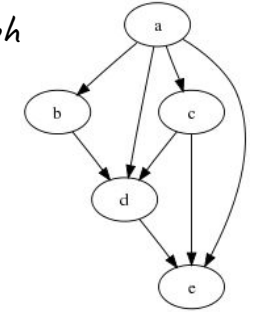
Data ed ora: informazioni relative al tempo di creazione ed ultima modifica del file

6) File System

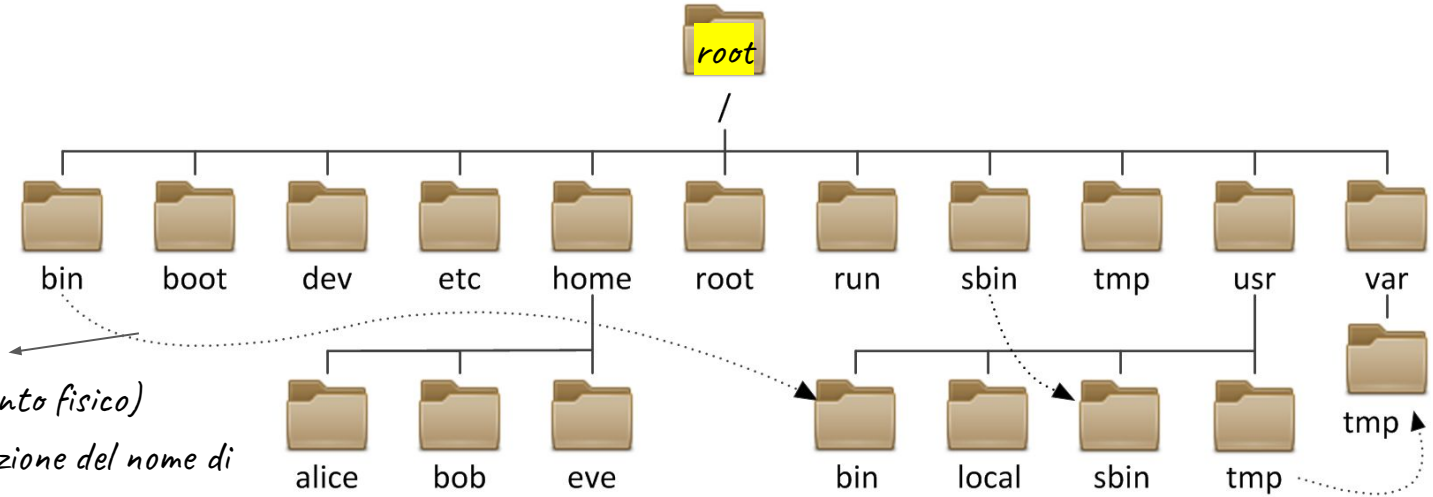


6) File System

Directed Acyclic Graph



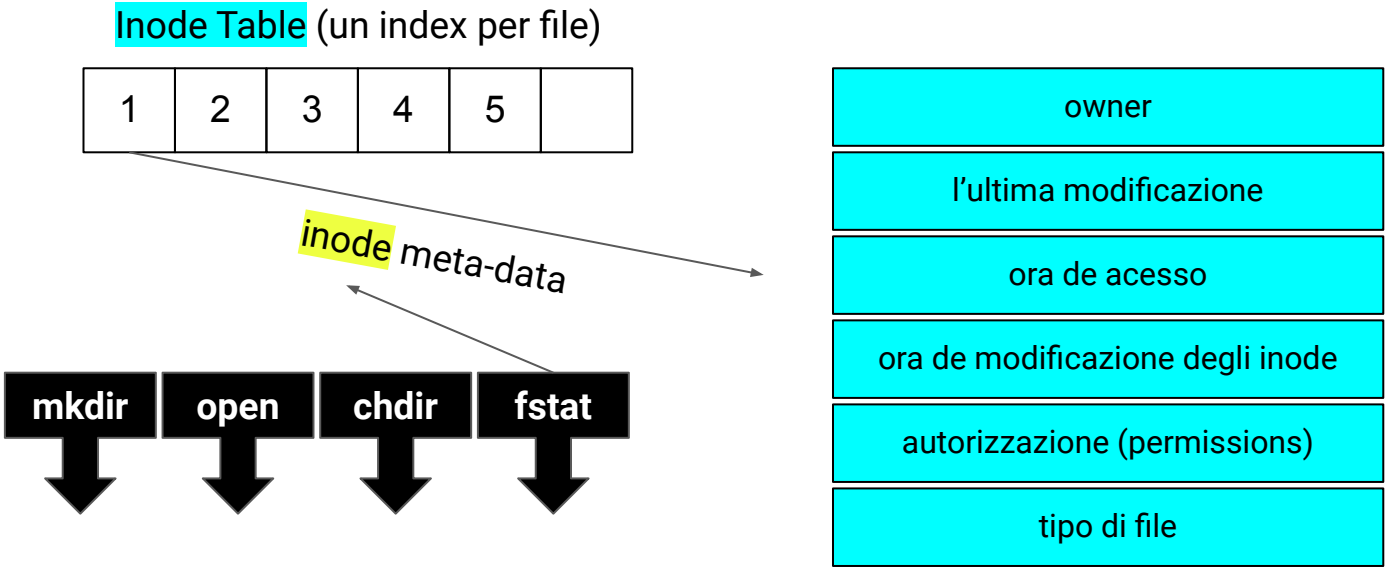
- Dal punto di vista del sistema operativo:
  - Files, sono Byte arrays
  - Directories, referenza a altri directories o byte arrays.
- I directories sono organizzati come uno alberto (tree -DAG)



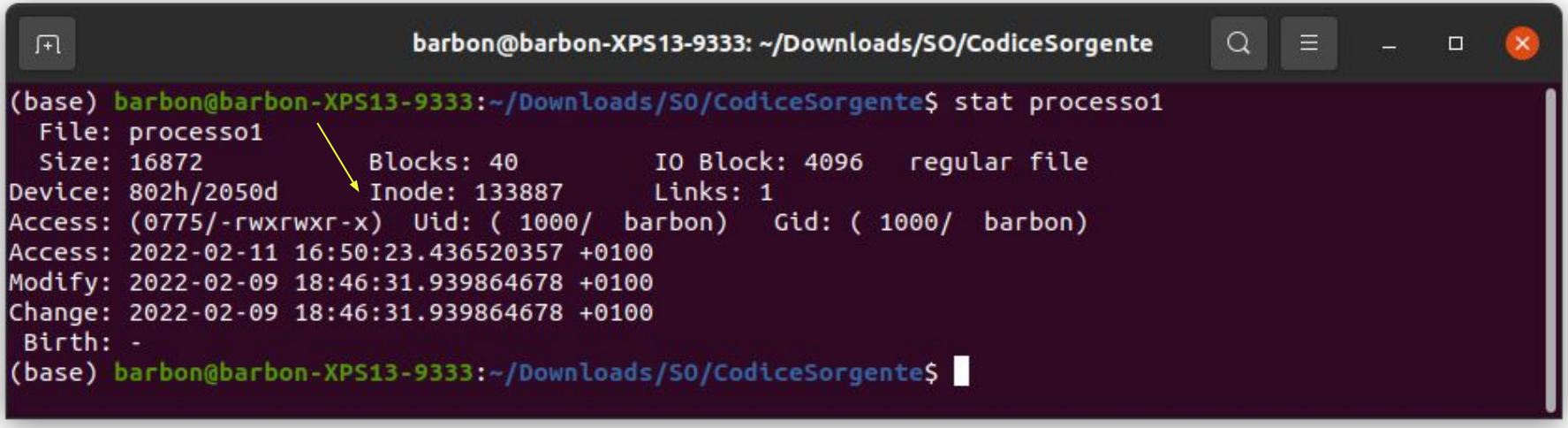
link: (collegamento fisico) indica l'associazione del nome di un file al suo contenuto.

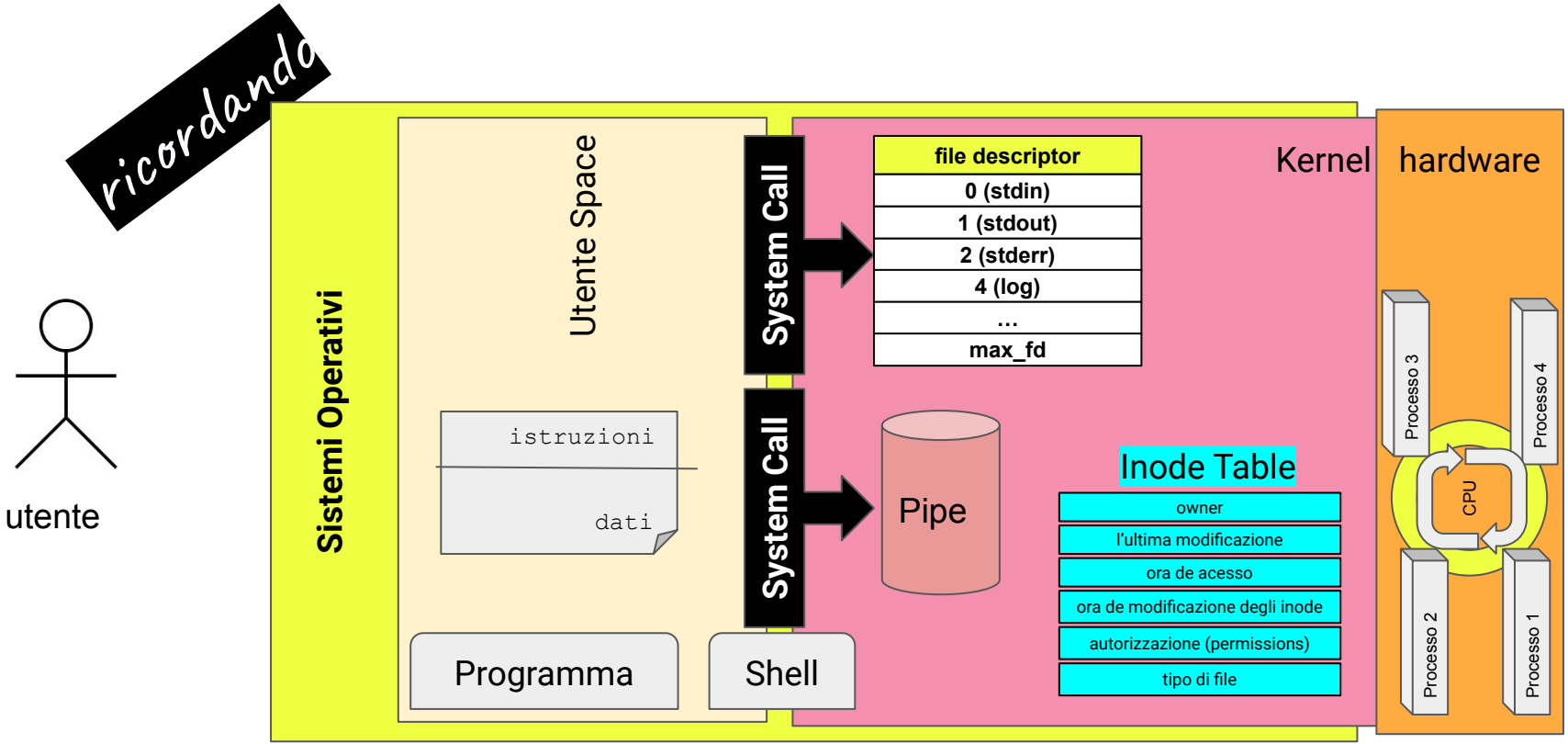
6) File System

- un inode è una struttura dati sul file system che archivia e descrive attributi base su file, directory o qualsiasi altro oggetto.



6) File System




ricordando
utente
Sistemi Operativi
Utente Space
System Call
System Call
file descriptor
0 (stdin)
1 (stdout)
2 (stderr)
4 (log)
...
max_fd
Kernel
hardware
istruzioni
dati
Pipe
Inode Table
owner
l'ultima modificazione
ora de acesso
ora de modificazione degli inode
autorizzazione (permissions)
tipo di file
Programma
Shell
Processo 3
Processo 4
CPU
Processo 2
Processo 1

Grazie!

Le referenze nel Moodle.