

# PROCESSI PTM

Un elaboratore esegue diverse operazioni → Ogni operaz.  
uso un ALGORITMO

ALGORITMO = INSIEME DI

ISTRUZIONI → INSTRUZIONI

per LAVORAZIONE DATI

ELABORAZIONE → BARRIERA → ALGORITMO

IN

LAVORAZ.

MANUTENZ.

Processo

PROVA UN PROGRAMMA VENIVI

ESEGUILO UNO VOLTA,

PER LA POTERNE DEL SISTEMA DIRETTIVI

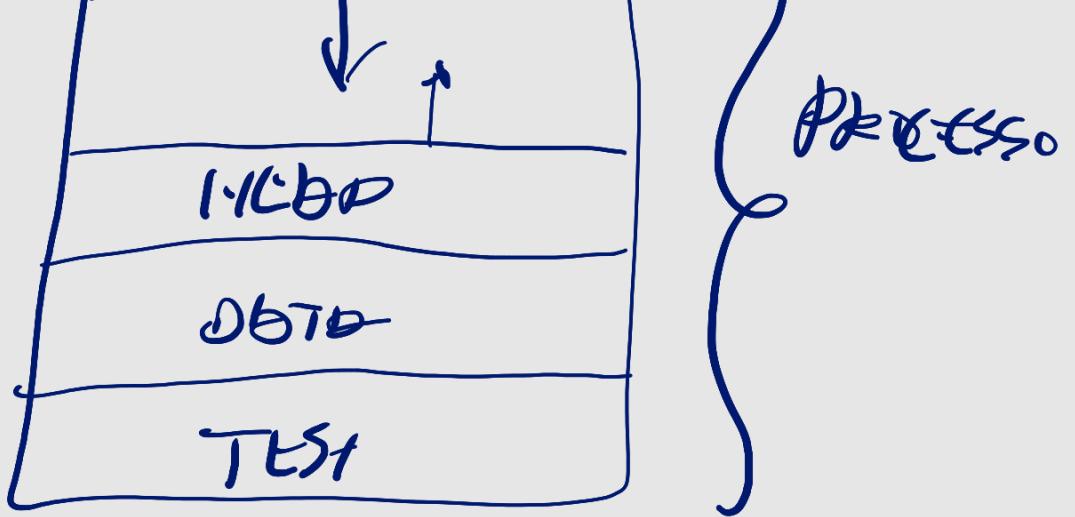
DI OPERA → PERMETTERE DI ESEGUIRE

PIÙ PROCESSI IN CONCETTO DI PARTE

→ SUPERVISORE DELLE RISORSI

SCHEDULING

STOCK



→ PID +

PRIMO PROCESSO → INIT → ACCES SIGHT PC → AVVIO DI UN PROCESSO IN



UTENTE → CREA process.

PROCESSI IN

BACKGROUND

SERVIZIO

↳ IL SO FUNGE A DIVIDERE SYSTEM SPACE

PROCESSI → PID

IN COMPUTER COMPLESSI → PROCESSI SUCCESSIONI,

STATE DI UN PROCESSO

↳ CREA (NUOVO) → PRIMO

{  
 STA (STANDE)  
 STA (ASPIRANDO)

→ TERMINA

NON CE SYSTEM CODE → PROCESS. NEGLIA UN ALTRO PROCESSO

↳ FINITO

PROBLEMI MIGLIORI → AL FILIO DI NON  
INTERESSO

## PROCESS CONTROL BLOCK

↳ POLEZIALE DEL PROCESSO.

- 1) STADIO DEL PROCESSO
- 3) PROGRAMMATORI → SUCCESSIVI  
    {  
        - ISTRUZIONI  
        - FETTORE - DECIDE
- 2) ATTIVAZIONE  
    (ES. → INTERRUZIONE  
    IL PROCESSO)
- 6) RISORSE  
    USATE
- 7) FILE ATTIVI /  
    OP IN ATTESO

SC. <  
PRENDE INIT

IN CUSTODIA

2) PID

PRIMA

3) SCHEDULAZ.

↳ Sincronizzatori

Processi

## SCHEDULAZ.

↳ QUALE PROCESSO ESTATEVICO DELL'ALTRA

→ CIOchè PROCESSI PRONTI C'È PROCESSI IN ATTESA

## CLASSIFICATIONE PROCESSI

BOUND

- 1) OP I/O → LIBERI  
    E SCARICATI DA CPU

ES: BLOCCATO  
RISOLTO DA IC

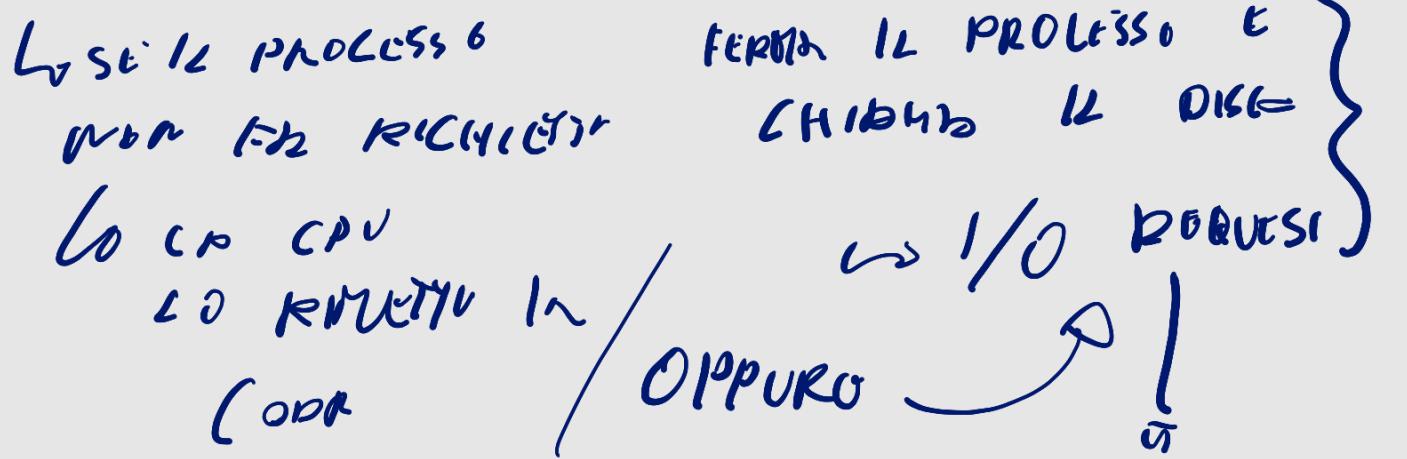
- 2) CPU BOUND  
    {  
        - OPERAZIONI CHE STAVANO  
        - IN CPU  
        - "PRIORITÀ"

## TIME SPLITTING

↳ ogni processo ha un tempo

## READY QUEUE (LISTA)

1) PROCESSI NERI = COSE DI PROCESSI



LE SOGLIE

→ PUNTO DI CODA VD

PROCESSO FERMO

IL PROCESSO

OGNI 100  
PACCHETTI  
FATTE ALLA  
VOLTA SERVITA

→ OPPURE PROCESSO IN

PUNTO

## SCHEDULER

→ delega a quale processo associare le CPU

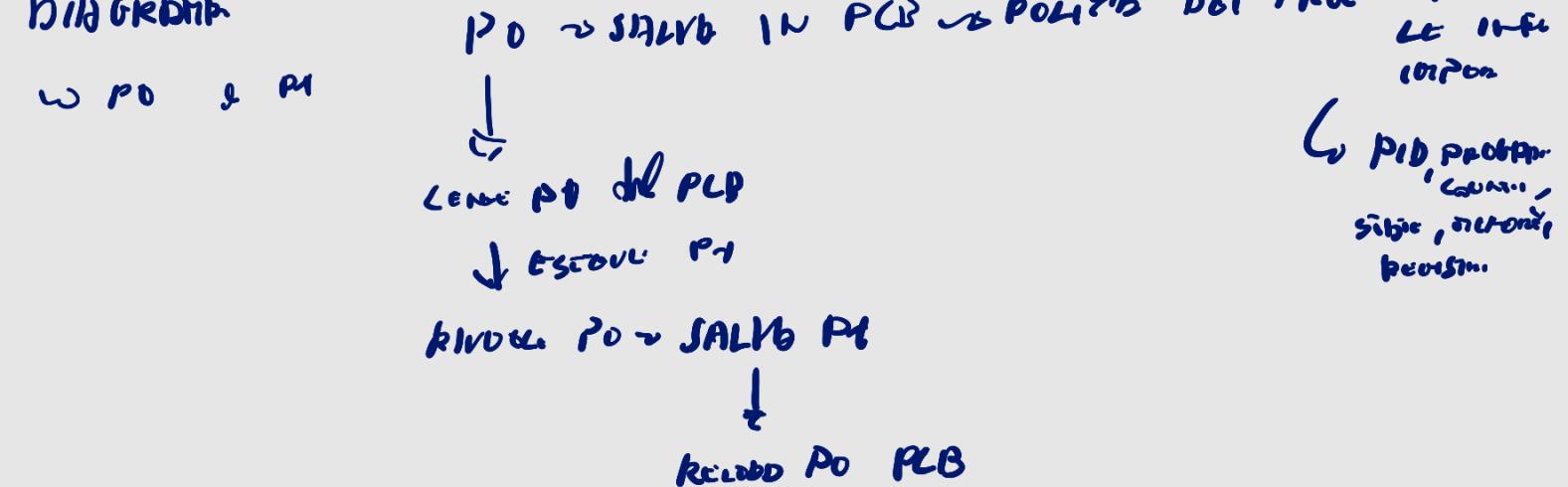
PENSARE ALLA SCELTA DEL PROCESSO

↳ IL PROCESSO DEVE SWICHARSI VEDENDO IL  
PROBLEMA QUINDI BISOGNA SOLVEDERE LO STATO → PIB

↳ COSÌ VENGONO CALCOLATI ANCHE QUELLO NUOVO

Stato Sbloccato → Ripristinato → CONTEXT SWITCHING

il tempo del context switching deve essere rapido → operazioni  
tecniche



↳ PID, PROGRAM  
'COSTRUZIONE'  
SUBIE, INVERSO,  
PERMISI.

## XED

- il processo ha finito il suo compito e lo dice
- ↳ KINDE
- il processo lo riconverte Downtime
- ↳ Lo reinserisce nei processi Attivi

## ALGORITMI SCHEDULING

- DYNAMIC: RIDURRE IL TEMPO OTTO DEL PROCESSO
- il SO ha sia il futuro di consumo del processo

## FIFO - FIRST COME FIRST SERVED

- primo processo che entra va **CPU**
- non parallelizza i processi
- processo intero viene fatto

## SHORTEST-JOB-FIRST SERVED

- priorità al più breve
- ↳ Non si sa quando dura un processo

## ROUND ROBIN

- subdivisione time (-time) nelles CPU
- ↳ tutti i processi uguali

## PRIORITY SCHEDULING

- processi con priorità più alta vengono eseguiti

↳ UN PROCESSO CON POLIS PRIORITÀ NON HA UNA OSAI

E SERVIZIO

## MULTI LEVEL QUEUE SCHEDULING

CODI DIVERSE A LIVELLO DI PRIORITÀ

SE SONO CODI IL SOLO DEGNAZIONE

RISOLVE IL PRIORITY SCHEDULING → PIÙ DI UNO CONCETTO

## GARANTEZI FAIR SCHEDULING (LINUX)

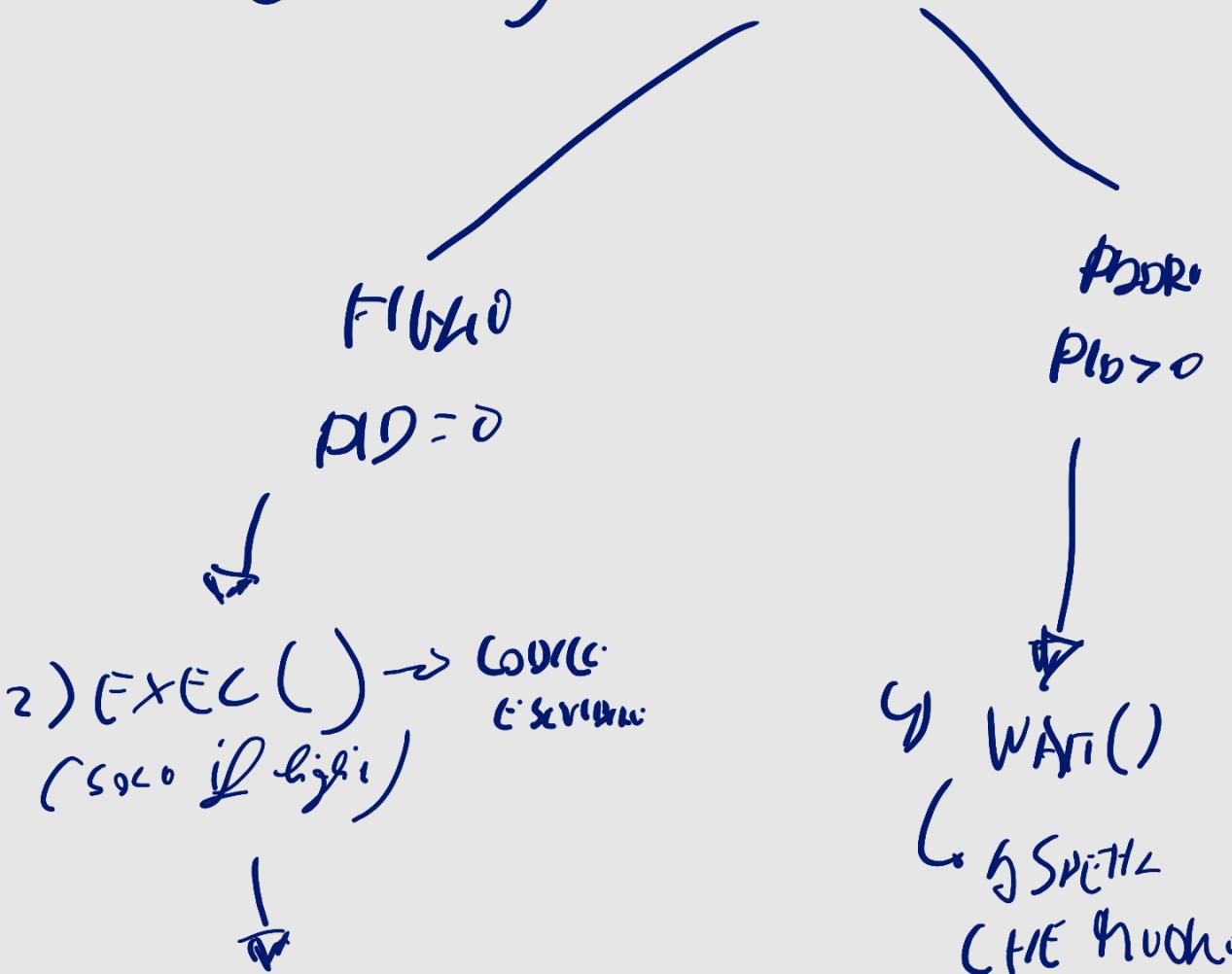
↳ UNI PROCESSI HA UNA POLICY ASSOCIAZIONE DELL'UTENTE

→ OTTIENI CODI → CLASSE DI PRIORITÀ

## PROCESSI PARENT DUE (6)

PROCESSI VENGONO CREATI DA UN UNICO PROCESSO

PID (ID PROCESSO) → 1) FORK → 2) DOPPIO PID → PROTEZIONE



3) EXRC()  
↳ il processo  
finisce

IL FIGLIO

5) KILL LO FIGLIOS  
F

6) SIGCHILD → estrae IL SCHEDA.

→ RITORNA UN PID (mamma)

PID\_T FORK()

↳ copia TUTTA LA MEMORIA PER  
PROCESSO PAPPA

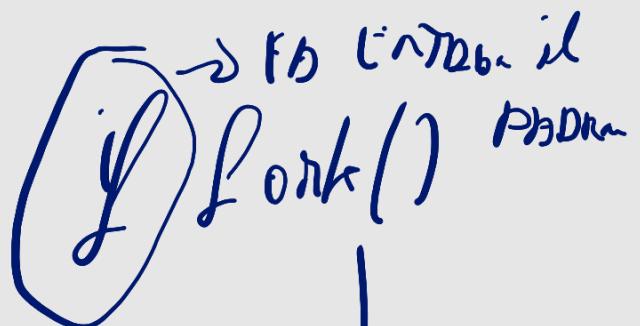
↳ (tutto, stack, bdm) IN FILE

ERRORE DELL'USO FORK → PID + EREDITAVI

(SOLI IL PAPPA)

CONTROLLARE CODICE

CO FORK() → 1) PAPPA



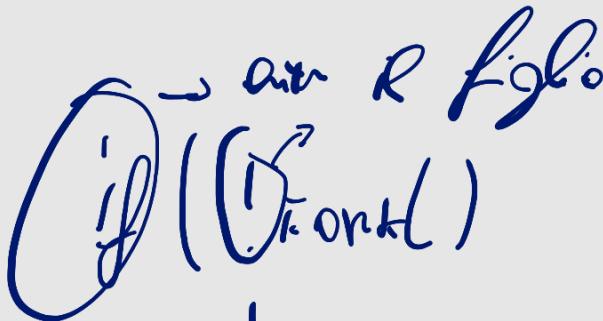
2) PAPPA

2) FIGLIO

1) PAPPA

2) PAPPA

2) FIGLIO



$\text{Fork}() \leftarrow !\text{fork}()$

1) phase:

2) prob

2) i. rando

—  
if ( $\text{fork}()$  ss ! $\text{fork}()$ )  
at & Curr at i. life

1) prob

2) curr

2) life

2) life

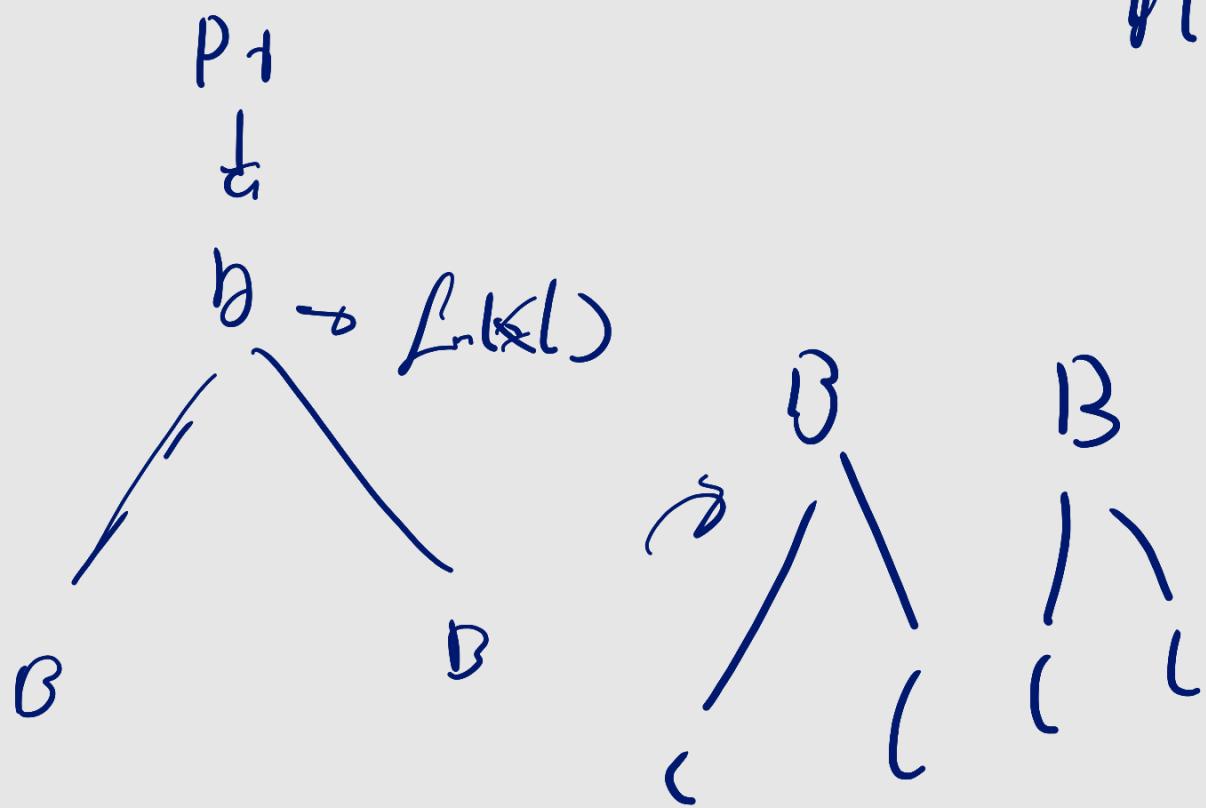
if ( $\text{fork}() \parallel \text{fork}()$ )  
prob 1:2

from F3

8

a force down

$f(fork() \parallel fork())$



**WAIT** → PID-T WAIT ( $\neq$  STB[0])  
 ↳ il figlio vuole → il padre ha già sede il  
 figlio richiede un edificio  $\downarrow$  SE

RITRONE

Lo PID DEL FIGLIO HA DIRETTAMENTE / O IN CASO DI ERRORE

{ PROC SENZA FIGLI: ERRORE  
 NON VIENE MESSO → PROBLEMI SUSPENDED VETRINO  
 SE I FIGLI NON SONO ANCORA MESSI  
 ↳ BLOCCO IL CHIAMANTE

1 figlio alla volta

il sistema operativo risponde con l'istruzione di FINE

**ZOMBIE** → PROC TERMINATO MA IL PDBE NON  
 HA ANCORA CHIAMATO ELIMINAR

**FRONT** → i h. vanno a pratica  
del podio → classica  
Iniz. uscita  
per noi p. g. **OPEN MARKET** **SELL PHASE**  
periodo di:

WATERPROOF - methyl methacrylate

{ PID = -1 → WAIT CLASSIC  
 PID > 0 → FIFO  
 PID = 0 → STESO GROUP ID (args)  
 PID < 1 → Right side GROUP ID = PID

GROU ID

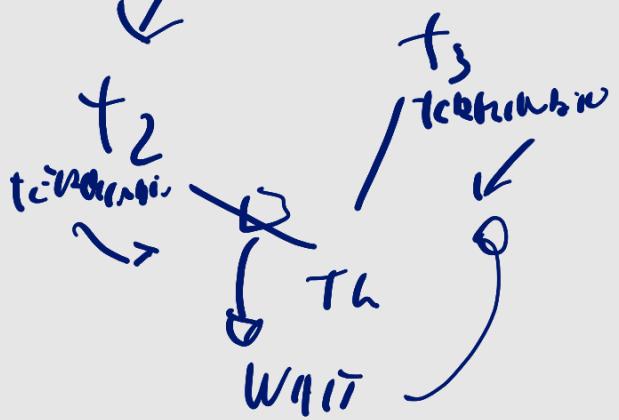
INTERI POSITIVI  $\rightarrow$  DEFINIRE GRUPPI DI PROCESSI

WAIT PID (PID, STATUS, Options) }  
 PID STATUS Options  
 STATUS ~ PID CHC DMRB  
 Options ~ from libcurl  
 L GETFILE IL DOWNLOAD  
 DCL PBDRC

WNOHANG → NON BLOCKING I/O FILEIO È POSSIBILE  
ED È MOLTO

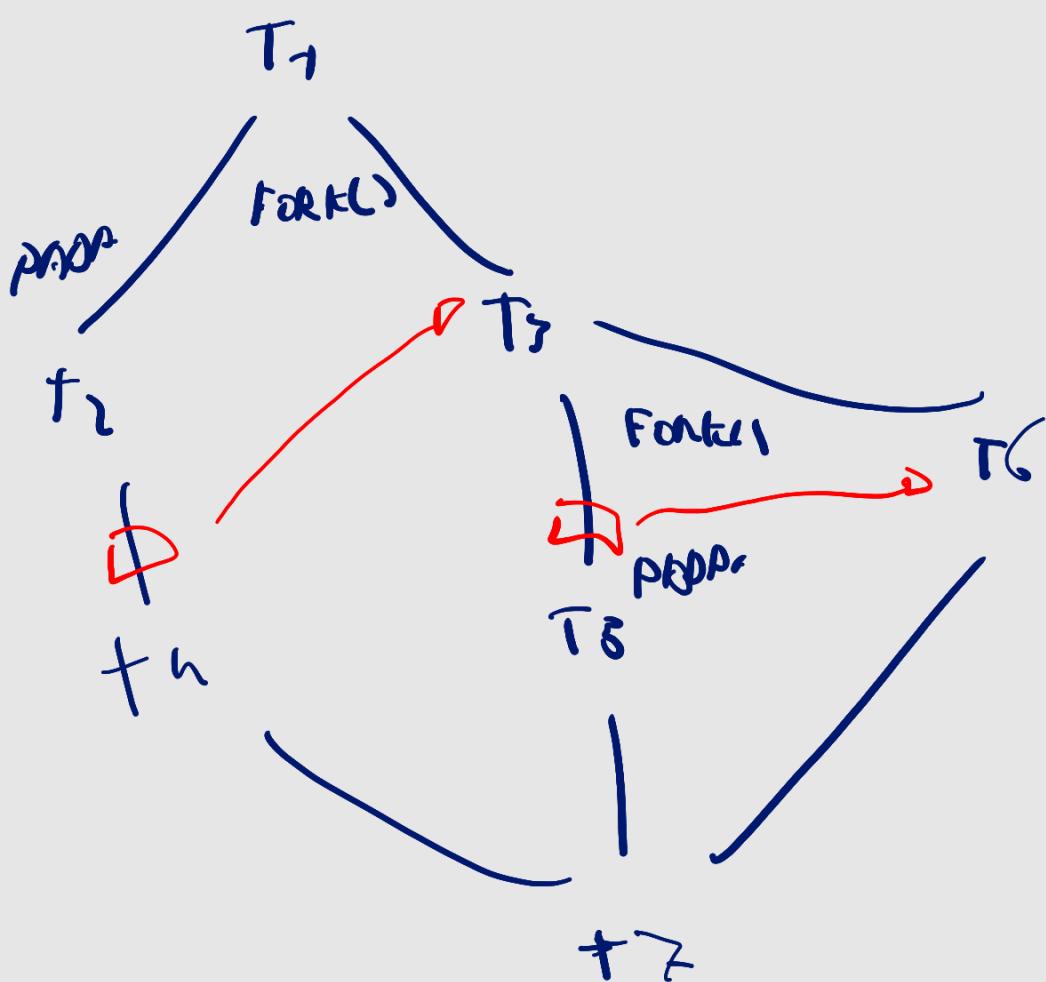
Trd  
= O<sup>2</sup>K(s)

# PROGRAMATOR POLSKA



$FORK(1 \rightarrow T_2 \text{ escurs. e } T_3 \text{ pure} \rightarrow T_2 \text{ fini wait}$   
 $WAIT( \text{int } * ) \circ \xrightarrow{\text{Escursione di }} FIGUR \quad \text{fini } T_3$

---



$\rightarrow$  si preferiscono ATTENDERE SOLO I FIGURI !!!

$EXEC()$

$\rightarrow$  QJPNB, RIUNO CHIOMBI UNO  $EXEC \rightarrow$  Come

PROCESSI RIMPIAZZANTI → TUTTI E' RICONTRASSO IL  
 → RIPRIETE  
 KPL MATH  
 (STESO PID) / LA EXEC CREA UN PROCESSO  
 YORE DI MEMORIA ADIOS (TEXT, DATA, HEAP, STACK)  
 DIRETTORE → VAR OR AMBIENTI PID  
 (PID)

NON VIENE CREATO(?) => FILE APERTI CON FILE CLOSE-DE-FILE

EXECVE SYSTEM CALL, LA EXEC CON "P" → RICEVONO IL PATH DELLSIGLIO  
 E NON IL PATH

EXEC(L)P ("CP", ...);  
 = EXECL ("USA/DIR/CP");  
 → SPECIFICARE GLI ARGOMENTI  
 → NEL TERMINALE VA A NELLE PATH SE C'È qualcosa  
 CORR "CP"  
 → CORR IN LIST  
 ("CP", 1, 2, 3, 4)  
 argv(0)(1)(2)(3)

se invece della L metto la "V" → argomenti tutti in UNICO VETTORE  
 La EXECV (\*PATH, \*argv[]); di PUNTATORI → PRIMA  
 METÀ

La EXEC CON "E" → VETTORE DI  
 VARIABILI DI  
 PONIBIZI  
 VETTORE  
 DI PUNTATORI  
 DI CHAR  
 "ENVs"

## Funzione exec

Esercizio: si scriva una semplice shell usando le funzioni fork, wait e exec

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#define MAXLINE 128
int main() {
    char buf[MAXLINE];
    pid_t pid;
    int status;
    printf("%% ");
    /* prompt */
    while (fgets(buf, MAXLINE, stdin) != NULL) {
        if (buf[strlen(buf) - 1] == '\n')
            buf[strlen(buf) - 1] = 0;
        if ((pid = fork()) < 0) {
            printf("errore di fork "); exit(1);
        } else if (pid == 0) {
            /* figlio */
            execvp(buf, buf);
            printf("non posso eseguire: %s\n", buf);
            exit(127);
        } else
            /* controllo */
            if ((pid = waitpid(pid, &status, 0)) < 0) /* padre */
                {printf("errore di waitpid"); exit(1);}
            printf("%% ");
    }
    exit(0);
}
```

*SPERAVO CO' DI PIÙ*

*Dove METTO CO' DI PIÙ*

*HO ENDO DA INPUT*

*NUOVO PIRETTORI*

*PATH*

*CONTROLLO*

*FU-LI*

Nota: per gestire gli argomenti dei comandi invocati, bisognerebbe manipolare le stringhe

## Funzione exec

Esercizio: si consideri il seguente programma.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (int argc, char ** argv) {
    char str[10];
    int n;
    n = atoi(argv[1]) - 1; → PARAMETRO DA STRIKER
    printf ("%d\n", n); → INT(0) (-1)
    if (n>0) {
        sprintf (str, "%d", n);
        execl (argv[0], argv[0], str, NULL);
    }
    printf ("End!\n");
    return 1;
}
```

*↳ NUOVO  
PROGRAMMA*

*↳ PASSO IL PATH E LA STR  
CONTIENE N*

#5

44

(1) (2) (3) (4) (5) (6)  
*↳ 4, 3, 2, 1, 0, End*

## SYSTEM → Non sistema CDE

Se invece un comando BASH è eseguito ha le seguenti conclusioni:  
 int system(*TERMINALI* & comando); → PROBLEMI CON LE FILE E  
*↳ FILE* E CON IL COMANDO ESSELENTO

## EXIT → UCCIDE IL PROCESSO

void EXIT (int status) → viene pulito il BUFFER  
 DI CONSOLE E FILE

→ EXIT → SYSTEM CALL → TERMINA SENZA COMANDO  
*DELC BUFFER*

→ fin di CLICCHERIO → fin PULIZIA E chiavi  
*da -EXIT*

ERRORE DI

TERMINAZIONE → ABORT()

i processi non hanno più memoria del processo.

EXIT → il kernel ricorda a terminare

(ls /proc / /proc/PID)

STRUCTURE Linux → INFO SUL PROCESSO → PROCS  
↳ DIRECTORY /proc/PID SYSTEM VIRTUAL  
/proc/PID/SIGNALS → info SU PROCESSO

{  
SIGINFO  
+ PID → THREAD GROUP ID  
PIDS  
PPID → PID PROCESSO PBOSS

→ {  
0 STDIN  
1 STDOUT  
2 STDERR

/proc/PID/FO → link a file gmt del processo

↳ /proc/TID/FO → STDIN  
↳ PROCESSO PID

SEGNALI → INTERRUPT → AWAKE SULLA CPU

INTERRUPT → thru RPLP CPU → STOPPED CHE DEVE  
fare qualcosa

{  
Hardware che notifica un evento  
System può chiamare (timer) o INTERRUPT  
SOFTWARE.  
("sono funziona  
CHE si avviano su INTERRUPT)

SEGNALI → come un INTERRUPT → da PERM  
AI PROCESSI  
da INTERRUTTORE

• INIZIATIVA → NOTIFIKA UN EVENTO  
UN PROCESSO → A PROCESSI

WENDETTI → { kernel per comunicare EVENTI  
UN ALTRU PROCESO → IN BASE  
A, PERMESSI

ALTRI SISTEMI HANNO UN MIGLIO IDRITTAZIONE → COME IL  
PID

↳ SIG

UN SEGNALE → { PUÒ ESSERE INTERRUZIONE  
TERMINARE IL PROCESSO  
SOSPENDERE IL PROCESSO

SIGKILL / SIGSTOP → IMMOBILI

SIGNAL HANDLER → LI FESTA SE MI ARRIVA UN  
DETERMINATO SEGNALE

VITÀ DI UN SEGNALE

CREATORE → KERNEL / ALTRI PROCESSO

DESTINAZIONE → STERMINARE IL PROCESSO

DESTINAZIONE → IL KERNEL FA ELIMINARE / TERMINARE

→ I SERVIZI NON SI ACCORDANO CON DIVERSI  
UN SEGUENTE SEGNALE

SIGACTION

→ ERRORE = -1

INT SIGACTION( ... )

... → CREA UN SERVIZIO

INT SIGNUM → SEGNAL NO VERTIKAL  
CONST STRUCT SIGACTION & SIG → FÜR ABLIEFUNG?  
STRUCT SIGACTION & SIGSET → CORRESPONDING PROCEDURE

SIG-HANDLEK {  
SIG-IGN → IGNORE  
SIG-DFL → DEFAULT}

SIG-MASK ← FORWARD DD BLOCKER  
→ SIG\_BLOCKSET (WÄHLER)

SIGALT → INTERRUPT

FORWARD FORWARD SIGALT:

SETUP, REVISORI PLEN SIGACTION

SECTION (SIG, SSA, 'H')  
↓  
Regist  
SIGALT

SIG-HANDLEK → SIG-IGN  
SIG-FLAG = 0 (DEFAULT)  
SIG\_BLOCKSET (S SIG-MASK)  
→ SIG-MASK

Regist  
Blocker

MY HANDLER → RECEIVE UN SEGMENT → INVOLVING  
OBJEKT SO

SIGACTION → INVOLVING OBJEKT RETRIEVE → SEARCH  
CONSUMED

PROCESS → EXECUTE → CHAIN OF SIGACTION

↓  
SIGACTION

↳ 1. PROZESS 1. KONTROLLE 1. SOSPENSION II. PROZESSO

BARRICARE IL SISTEMA DA INTERRUZIONI  
SIGTERM → FD PRIORITY & HANDLER  
↳ Handler ha priorità  
↳ ritorna a funzione  
di process

SIGNAL → SYSTEM CALL BOSS<sup>6</sup> Linux EXPORTED  
SIGHANDLER + SIGNAL (INT SIG, SIGHANDLER HANDLER)  
↳ system → punt  
be utente A FUN

KILL SYSTEM CALL (PID-T PID, INT SIG) ↳ CHE SCHIENE  
HAND?

{ mandare un segnale SO UN PROCESSO / UNRUPT  
PID SO → SPEDISCE A PID  
=D → tutti i processi con = UNRUPT KILL  
-+ → UNTII

Root può mandare segnali a processi che non lui  
mettre in processo può mandare un segnale solo a  
processi che stesso utile

POLICE → SY → inviare un segnale a se stesso  
↳ come kill (GETPID(), SIG)

SYSTEM CALL PAUSE

↳ sospende il processo fino all'arrivo di

VR SELVAGGIO → RETROVR →

SYSTEM CALL  $\text{sleep} \rightarrow$  sleep (seconds)

time-out creato dal KERNEL  
↳ il tempo effettivo  $\text{only seconds}$   
è causa del tempo di reazione del kernel  
più breve aumentato

mbut SLEEP

## INTER-PROCESS COMMUNICATION

→ i processi sono in esecuzione  
contemporaneamente

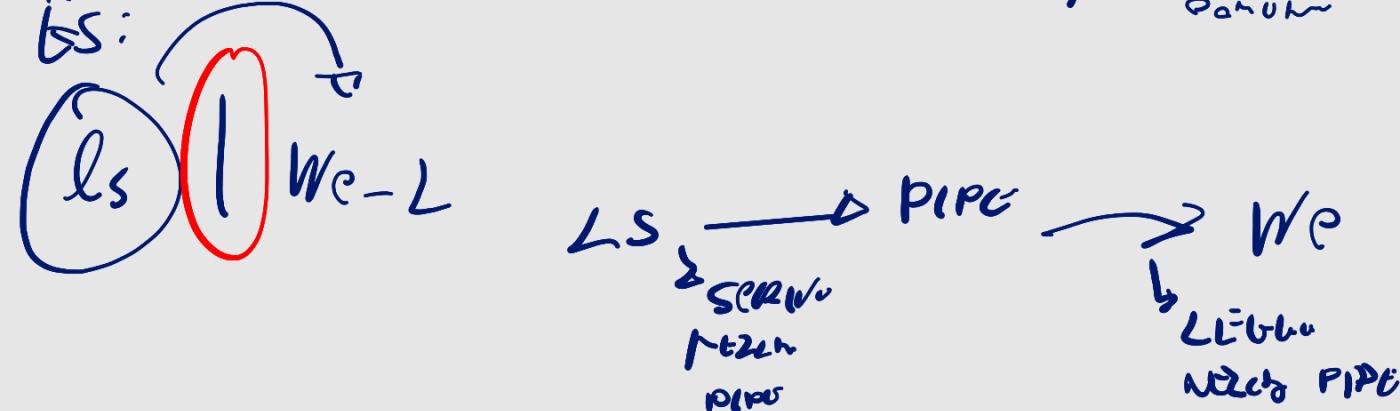
{ INDEPENDENCE → per i punti che  
COOPERATION → interazioni con altri processi

IPC { Segnali di messaggio (secondi)  
      Segnali bloccanti PIPE / FIFO  
      Memoria Condivisa MMAP

Pipe → scrivere dati → SYSTEM CALL  
READ WRITE  
PRODUTTORE - CONSUTTORE

→ SISTEMA IN MEMORIA

→ COMUNICAZIONE SOLO SENSO →  $P_1 \xrightarrow{\bullet} P_2$   
↳ processi con qualsiasi



( $\sigma$  PIPE Lifetime ( $\sigma$ ) sickens  $\rightarrow$  BITE STRENGTH

→ Plausibilität einsetzen

→ Plausibilität  
→ o. g. in LECTURZ

INT PIPE (int ErrorCode[2]) 1 111 11 SCHUTZEN

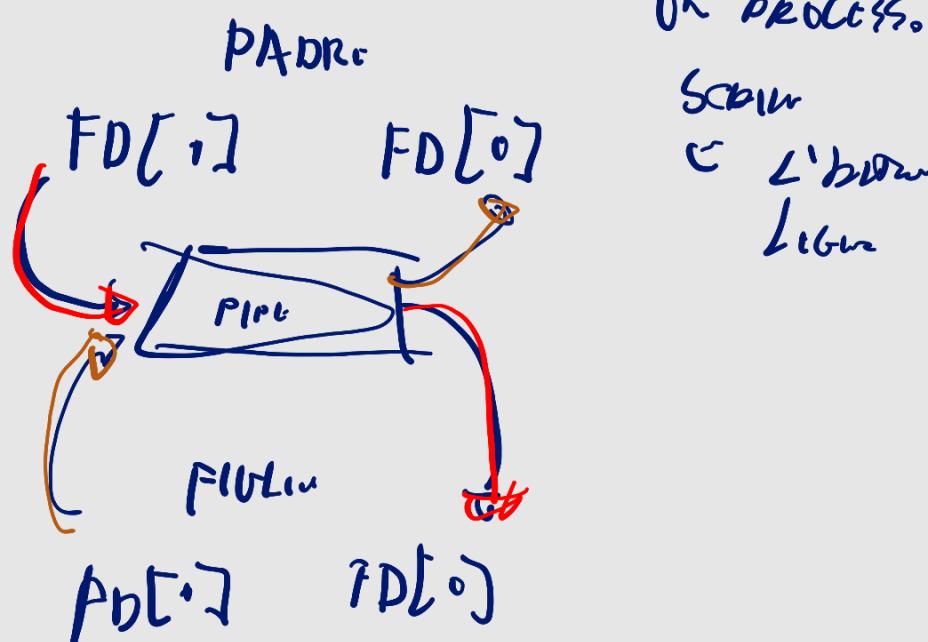
Rebo → threat for you nor come action on BFM.

*WRTIO* → *u b p<sub>T</sub>* pt é pmais → WRTIO Blocoam

↳ ze River Dog norm file bravo in Session → P1P1 norm

## PIPE TRAP PIÙ PROCESSI

$P_{\text{phase}} \rightarrow \text{var } L_{\text{PIPO}} \rightarrow \text{FD}[6] \times [1]$



FIFO & PROCESSI CONCERNANTI AL CONSUMO

↳ TIPO DI FILE & S-LS FIFO

→ O come ritorno  
se la success

MR FIFO → (PATH, MODE-T MODE)

↳ PERMESSI CEDUTI SUL FILE

OPERATORI FILE

{  
OPE → RAPPRESENTAZIONE - CONSULTAZIONE PRESENZA  
LETTERA - SCRITTURA  
F OPEN  
↳ STAVO SI BLOCC  
OPERA SI APREN

→ utenze di FILE SYSTEM

Ex non  
particol  
DBB

O - NON BLOCK

→ NO  
SEARCH  
LOCATE  
NOTICE

DATI FIFO → MIGRAZIONE DELLE KUNZE

