GESTIONE PROCESSI

(OVVERO ADDESTRAMENTO X FUTURI KILLER)

Pietro "Peterlaw" Virgilio==>pietro.peterlaw@gmail.com

SIAMO USER O SYSTEM ADMIN?

OBBIETTIVI DI MISSIONE

- Lo scopo di un buon System Admin è avere il pieno controllo di ciò che la macchina stà eseguendo
- Appena terminato il bootstrap il sistema operativo esegue svariate decine di programmi
- Quindi il gioco consiste nell'eseguire solo ciò che ci serve e UCCIDERE tutto ciò che è causa di problemi
- Come facciamo a capire cosa il computer stà eseguendo e come agire?
-

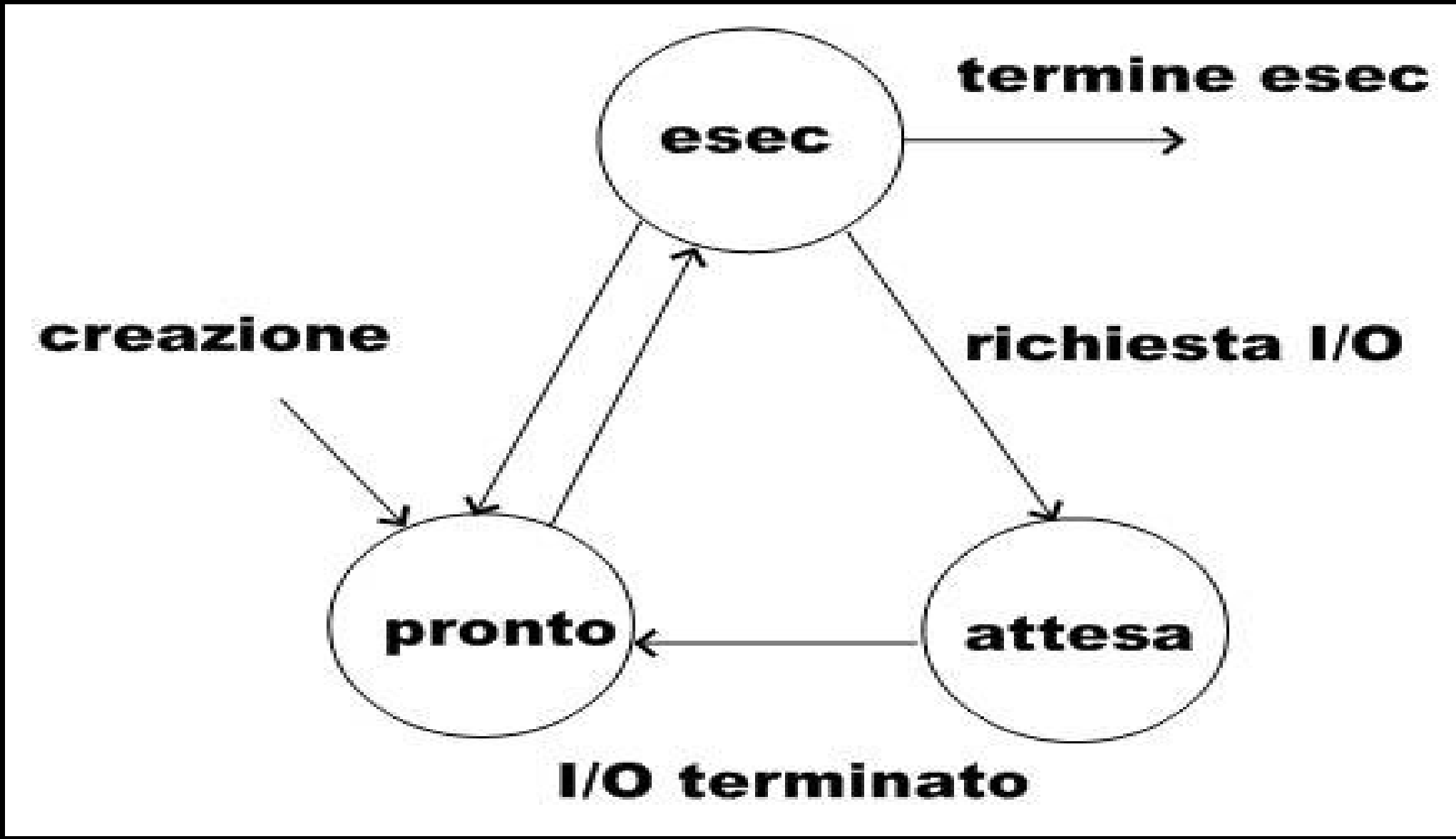
IL CONCETTO DI PROCESSO

CONOSCERE IL NOSTRO AMICO/NEMICO

- Un processo è un istanza di un programma in esecuzione su una macchina
- Ogni processo è generato come figlio di un altro processo detto padre
- L'unico processo senza padre in Linux è Init
- Il processo può eseguire delle chiamate di sistema o “syscalls” ad esempio per accedere a dei file o interagire con le periferiche di I/O

IL KERNEL

IL SEMAFORO DEI PROCESSI



STATI DEI PROCESSI

VIVO O MORTO,X

- **Runnable** : il processo può essere eseguito
- **Sleeping** : il processo è in attesa di una azione (tipicamente un I/O)
- **Zombie** : il processo tenta di terminare ma non è ancora stato deallocato
- **Stopped** : il processo è sospeso e non può essere eseguito

ATTRIBUTI DEI PROCESSI

FAVORISCA I DOCUMENTI

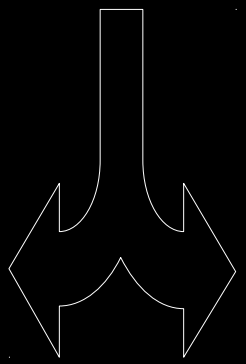
- **PID** : il Process IDentifier è un'identificatore numerico univoco del processo
- **PPID** : il Process IDentifier del processo padre
- **UID** : User IDentifier è l'identificatore del proprietario del processo
- **GID** : Group IDentifier è l'identificatore del gruppo a cui quel processo appartiene
- **COMMAND** : la riga di comando con cui il processo è stato lanciato che comprende anche i parametri ad esso passati
- **NICE** : è un valore numerico relativo che và da -20 a 19, minore è il suo valore e maggiore sarà la priorità che il kernel del so gli darà in fase di time preemption
- Inoltre tramite i tools di controllo che vedremo in seguito ad ogni processo è possibile associare quantitativamente le risorse di sistema che esso stà utilizzando (es ram,cpu,swap.....)

I SEGNALI NON STRADALI

IL NOSTRO ARSENALE

- Un modo che abbiamo per interagire con i processi è quello di inviargli segnali
- Un segnale è un valore numerico intero spedito da un processo verso un altro
- L'invio di un segnale ad un processo può causare 2 casi

Il processo ha una routine x gestire quel segnale



Il processo non conosce il segnale e delega al kernel la decisione

I SEGNALI NON STRADALI

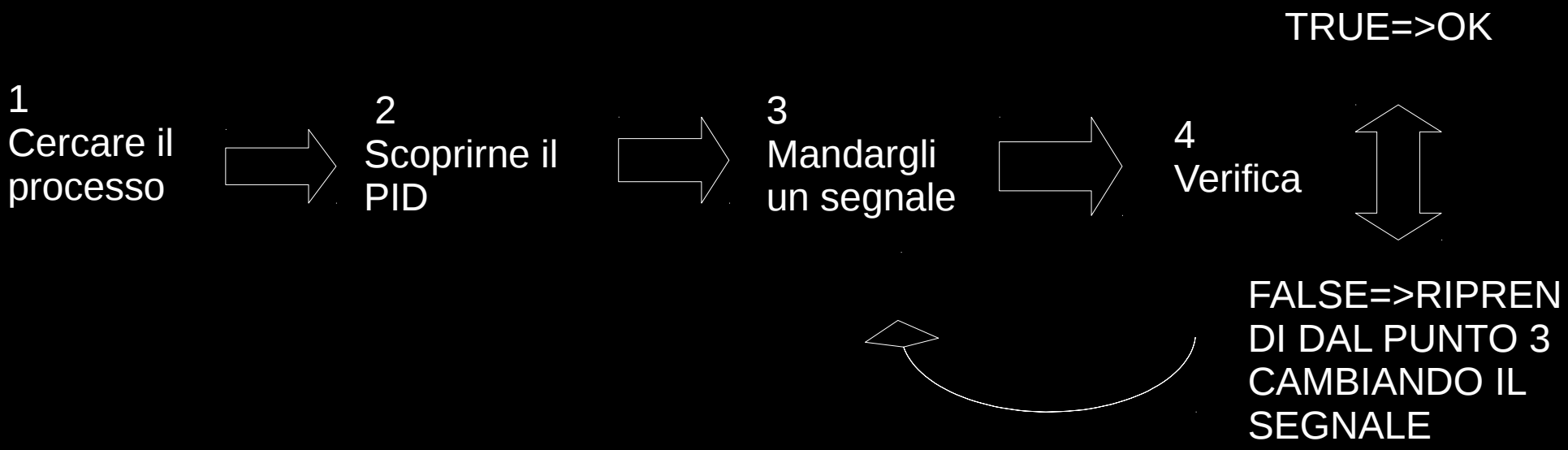
IL NOSTRO ARSENALE

- **SIGINT(2)** : Questo è il segnale che inviamo quando durante l'esecuzione di un processo digitiamo Ctrl+c, nella maggior parte dei casi termina l'esecuzione=Pistola automatica
- **SIGTERM(15)** : Uguale al precedente ma non ha una combinazione di tasti associata=Pistola semi-automatica
- **SIGSTOP** : Blocca l'esecuzione del programma in uno stato di attesa fin quando non riceve un segnale di sblocco= Teser
- **SIGCONT** : Riattiva l'esecuzione di un programma bloccato con la SIGSTOP=Una pera di Adrenalina
- **SIGKILL(9)** : Termina l'esecuzione del processo in modo insindacabile e senza la possibilità di una routine che lo gestisca=Missile nucleare perforante
- Inoltre i processi possono anche generare dei segnali denominati codici di uscita, in genere un processo che termina restituisce 0 segnalando che tutto è andato a buon fine. In caso contrario ci possono essere stati degli errori in fase di esecuzione

TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **Premessa** : In generale gestire i processi vuol dire usare la shell attenendosi ad un iter che nella maggior parte dei casi ci consente di ottenere il risultato cercato



TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **PS** : ci permette di vedere tutti i processi in esecuzione su una macchina (staticamente) passandogli opportuni parametri

  ps -e -o pid,ppid,state,command

  combinato con pipe e grep diventa uno strumento di individuazione utile ma macchinoso

TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **PS AUX** : è sempre ps ma con l'opzione aux che fornisce molte più informazioni utili come ad esempio la cpu e la ram
- **FREE** : ci dice quanta memoria ram è utilizzata utile se vogliamo fare una selezione per risorse *free -m*
- **SWAPON/OFF** : ci permette di conoscere le partizioni di swap disponibili, e di attivarle/disattivarle *swapon -s (-a.....)*

TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **TOP** : simile a ps ma l'output si aggiorna in tempo reale (dinamicamente), inoltre premendo h compare un menù integrato per gestire i processi e fornisce alcuni indicatori sullo stato complessivo del sistema. Infine è possibile visualizzare i processi di un singolo utente, utile per i server multiutente

  top -u “utente”

TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **HTOP** : ancora più avanzato rispetto a top, ha una toolbar per l'invio automatico di segnali ai processi, ha un opzione per visualizzare l'albero dei processi ed è a colori!

  La fregatura è che non è installato di default in nessuna distro quindi per es sulle debian-based

  sudo apt-get install htop

  htop

TOOLS DI MONITORAGGIO

ENTRIAMO IN AZIONE

- **KILL** : è il metodo per mandare segnali diretti al processo di cui viene passato il PID come parametro (il nome lascia intendere il suo scopo ;-)

  kill -segnale “PID”

  il segnale può essere usato sia nella sua versione numerica che letterale

- **KILLALL** : come sopra ma manda il segnale a tutti i processi con un certo nome decisamente utile per fare stragi di massa XD

  killall “programma” -segnale

PROCESSI E PRIORITÀ

METTETEVI IN FILA E NON SPINGETE

- L'algoritmo LRU che usa il kernel linux per l'assegnamento delle priorità di esecuzione tra processi è piuttosto efficiente ma a volte risulta utile interporsi ad esso per dare maggiori privilegi ad un processo di nostra scelta. Per riuscirci basta modificare l'attributo nice del processo a noi noto e il gioco e fatto.

PROCESSI E PRIORITÀ

METTETEVI IN FILA E NON SPINGETE

- **RENICE** : reinposta l'attributo nice del processo indicato dal suo PID

  renice “nuovo_valore” “PID”

- **NICE** : permette l'esecuzione di un programma con un valore di nice passato come paramentro

  nice -n “numero” “pathname”

ANALISI DELLE SYSCALL

CHE È SUCCESSO AL PROCESSO?

- Ogni processo può far uso delle oltre 400 chiamate di sistema messe a disposizione da Gnu/Linux
- Non sempre tali chiamate vanno a buon fine e questo genera malfunzionamenti di vario tipo..
- A questo punto risulta utile scoprire quali e quante chiamate di sistema un processo effettua.....

ANALISI DELLE SYSCALL

CHE È SUCCESSO AL PROCESSO?

- **STRACE** : permette di interporsi tra il processo e il sistema operativo scoprendo le syscall effettuate e risalire al presunto problema

  nb bisogna essere root per ascoltare le chiamate di sistema tra esso e il processo

  sudo strace -p “PID”

- In più è possibile lanciare un programma e ascoltare direttamente

  sudo strace “programma”

IL FILESYSTEM DEI PROCESSI

THERE IS NO PLACE LIKE /PROC

- All'interno di questa directory troviamo tutto ciò che riguarda i processi e il sistema
- /proc/cpuinfo => info cpu
- /proc/meminfo=> info memoria
- /proc/versione=> versione del kernel
- Le sottodirectory di /proc sono nominate come il PID del processo a cui sono associate e all'interno troviamo svariate info tra cui la cartella fd con all'interno i File Descriptor dei file aperti da quel processo

TUTTO È UN FILE

FAMMI VEDERE CHE HAI IN MANO

- La filosofia di Linux è “tutto è un file” che sia di testo che sia binario che sia un interfaccia con una periferica aprire file è una delle operazioni più comuni dei processi
- Sarebbe utilissimo sapere che file ha aperto un determinato processo
- **LSOF** : mostra una lista di file aperti da un processo passato per parametro

  lsof -p “PID”

DOMANDE?

SE TUTTO QUESTO NON DOVESSE FUNZIONARE RIVOLGETEVI A FAT ED

GRAZIE X L'ATTENZIONE!

```
$ killall peterlaw -9
```