Unix

Enrico Girardi

Login

Identificazione

Nome utente

Verifica esistenza utente

Autenticazione

Password / similia

Password, pin, badge, impronta digitale, impronta dell'iride e/o combinazioni di queste

Autorizzazione

Livello di accesso

Una volta entrato, ciascun utente può fare o non fare determinate azioni o leggere o non leggere (modificare, leggere, scrivere) determinati file

Classificazione Utenti

Superutente

UID: 0

Username = root

Può fare tutto

Utenti di sistema

UID: da 1 a 999

Username = nome del servizio

Non hanno shell

Accedono solamente a specifiche risorse

Utenti umani

UID: da 1000 a 2^64 (-1)

Possono elevare i loro privilegi

Hanno una shell

Home directory

Privilegi di Accesso

LETTURA	SCRITTURA	ESECUZIONE
r	w	x

s bit setuid/setgid

Autorizzazioni FILEs (Rappresentaz. TESTUALE)

Utente Creatore	Gruppo Ut. Creatore	Tutti gli altri Utenti
U	**g**	**o**
Utente che ha **creato** il file	**Gruppo (primario)** dell'utente che ha **creato** il file	**Tutti gli altri** utenti del sistema

A (all) = tutti gli utenti di sistema

Autorizzazioni FILEs (Rappresentaz. OTTALE)

0	Nessun Permesso
4	Permesso di **Lettura**
2	Permesso di **Scrittura**
1	Permesso di **Esecuzione**

===> E LORO SOMME:

Autorizzazioni FILEs (Rappresentaz. OTTALE)

7 (rwx)	Permesso di **Lettura, srittura, esecuzione**
6 (rw)	Permesso di **Lettura, scrittura**
5 (rx)	Permesso di **Lettura, esecuzione**
4 (r)	Permesso di **Lettura**
2 (w)	Permesso di **Scrittura**
1 (x)	Permesso di **Esecuzione**
0 (-)	Nessun Permesso

+ Setuid/ Setgid: prima cifra del numero Ottale

0	Nessuna impostazione
1	E’ impostato lo **sticky bit**
2	E’ impostato il bit setgid
4	E’ impostato il bit **setuid**

Set user id: permette di eseguire un file (eseguibile) con permessi del proprietario anzichè i propri

Sticky bit: i files o directory con sticky bit possono essere cancellati solo dal proprietario (tipico in cartelle di temporanee o condivise)

Sistema Posizionale

Prima Cifra	Seconda Cifra	Terza Cifra	Quarta Cifra
setuid/setgid	Utente che ha **creato** il file	Utenti che appartengono al **gruppo** dell'utente che ha creato il file	Tutti gli **altri** utenti

Tipi di FIles

-	File Regolare (file di testo)
d	Directory (cartella)
b	Dispositivo a blocchi
c	Dispositivo a caratteri
l	Collegamento simbolico
p	Named pipe
s	Socket Unix

⇒ Autorizzazioni FILEs (Rappresentaz. OTTALE)

esempio:

UTENTE CREATORE	UTENTI NELLO STESSO GRUPPO DEL CREATORE	ALTRI UTENTI
RWX	RX	RX
7	5	5

Credenziali di un Processo

Identificatore Gruppo Primario

Dell'utente che ha lanciato il processo (primary group identifier)

Identificatore Utente

Che ha lanciato il rocesso

Identificatori dei Gruppi Secondari

Dell'utente che ha lanciato il processo (secondary group identifier)

IDENTIFICATORI DEI PROCESSI

Ogni volta che un processo chiede accesso ad una risorsa, vengono confrontate le sue credenziali di accesso di questi 3 tipi,
E **per ciascun punt**o del percorso assoluto di un file :

Es: /
/etc
/etc/passwd

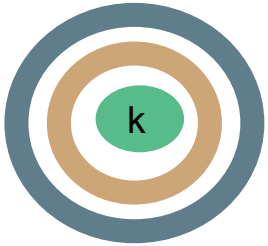
Vengono verificati gli identificatori per ciascun punto dall'alto (/ rot) in basso

Sistema Operativo

Software che gestisce hardware e software di tutta la macchina.

Sistema Operativo

Stratificazione "A CIPOLLA"



Kernel

- Driver dei dispositivi Gestore I/O
- Gestore dei Processi
- Gestore del file system
- Gestore della memoria
- IPC (Inter-process communication)

Sistema base

- Librerie di sistema
- Caricatore dinamico
- Sistema di init
- Comandi di sistema
- Shell
- Terminale

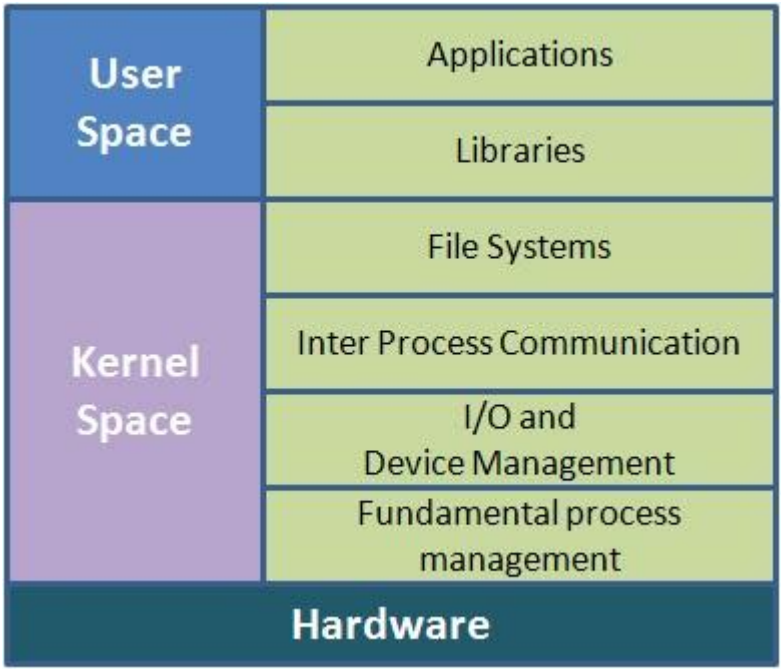
Programmi applicativi

- Compilatori
- Interpreti
- Ambiente grafico
- Suite di ufficio
- Browser
- Client e-mail
- [...]

SO

====> DIVERSI MODELLI

Kernel Monolitico (Macro Kernel)



- Tutti i **servizi** vengono eseguito in **kernel mode**
- Tutte le **applicazioni** vengono eseguite in **usermode**

Bottom-UP

Ogni strato implementa le proprie funzionalità basandosi sulle funzioni degli strati inferiori

Variante (LINUX): MACRO KERNEL CON MODULI

⇒ uso di moduli caricabili
⇒ il kernel diminuisce di dimensione
⇒ l'eventuale crash di un modulo non manda in crash tutto il sistema

Linux:
scritto in C (uso di puntatori alle risorse)
+ assembly per architettura

Micro Kernel

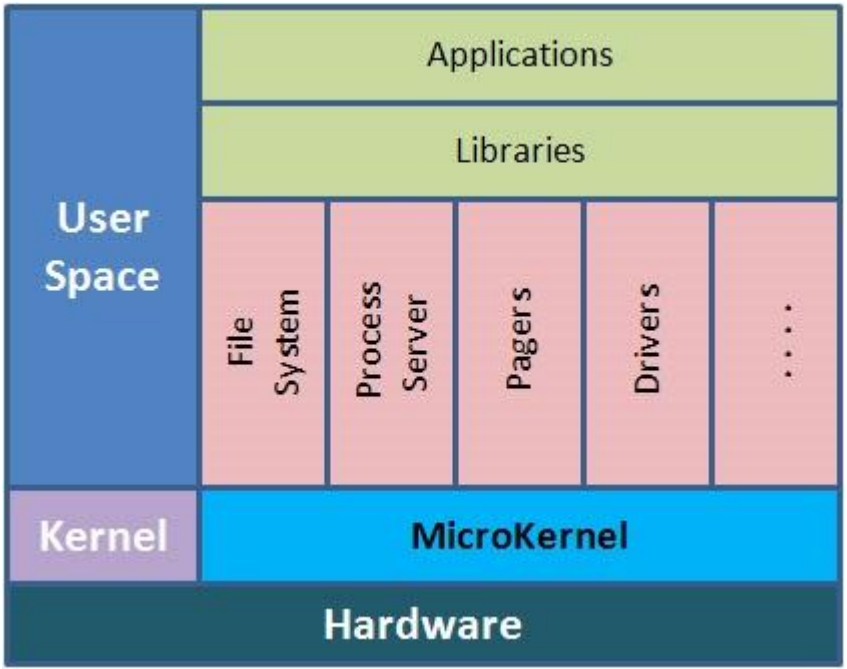
- Il **kernel** esegue solo **servizi essenziali**
- Il resto è demandato ad un **sistema (server) applicativo**

⇒ sistema di comunicazione a Messaggi

- SO robusto rispetto ai crash
- Kernel ridotto

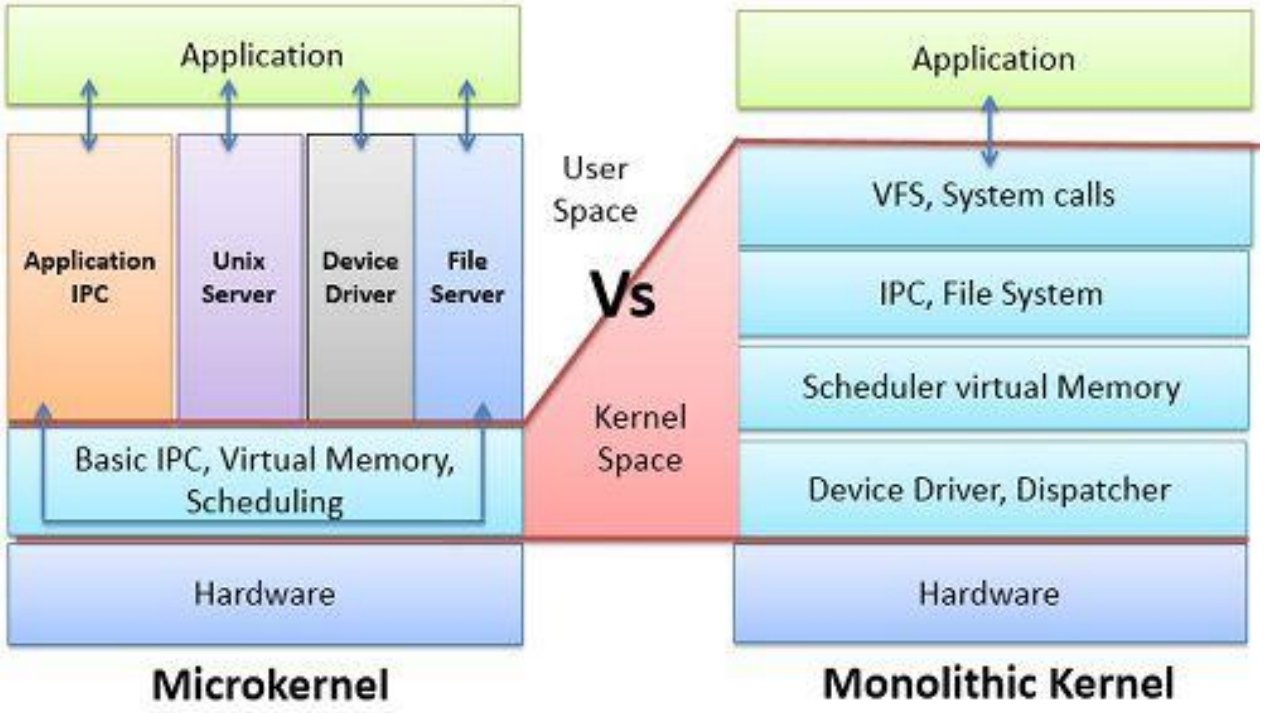
- Meno performante
- Per ogni chiamata di sistema ed ogni messaggi di scambio, c'è una commutazione user => kernel

Mach kernel, Minix

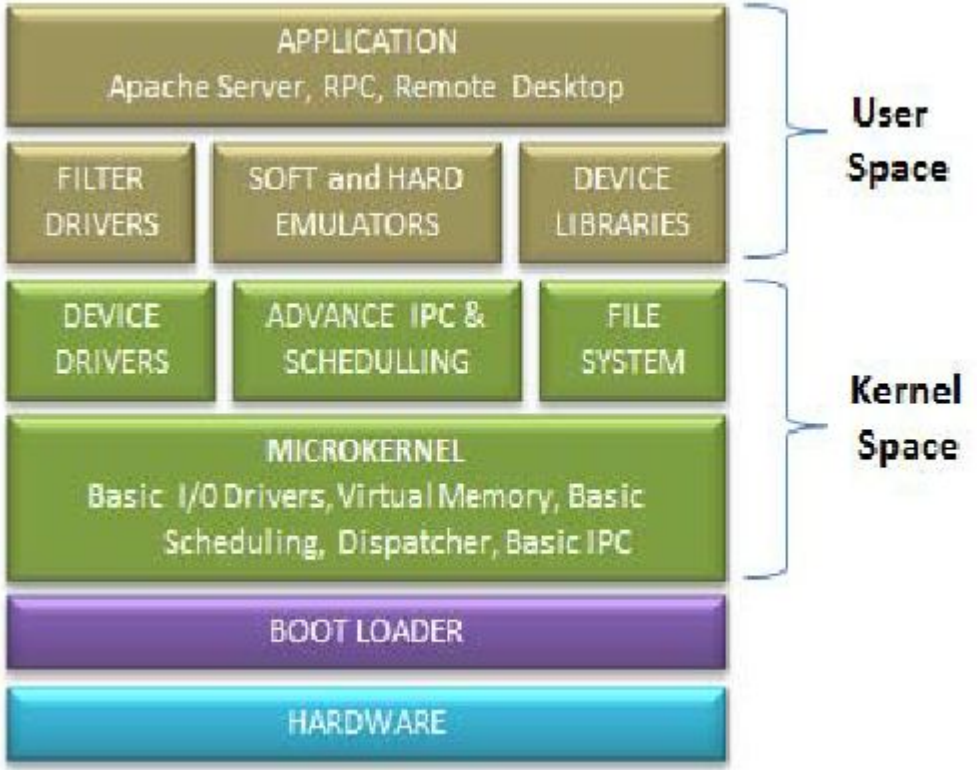


Differenze fra kernel Monolitico e Micro Kernel

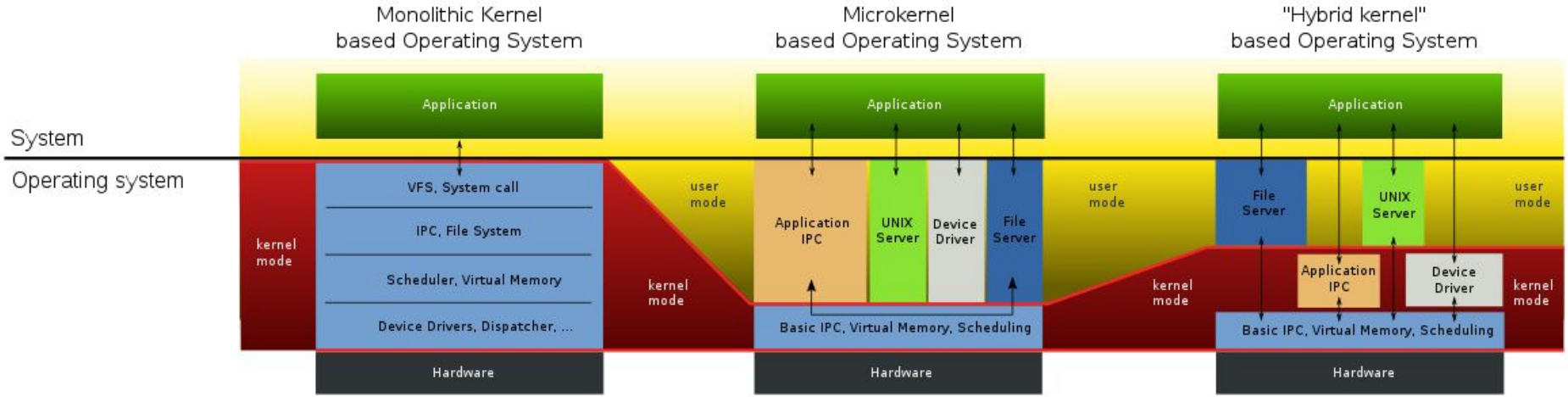
Schema grafico



Hybrid Kernel



A confronto


Monolithic Kernel based Operating System
Microkernel based Operating System
"Hybrid kernel" based Operating System
System
Operating system
Application
VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
kernel mode
user mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling

Buffering

I/O diretto

Oppure

I/O Bufferizzato

Svuotamento Buffer

BUFFER APPLICATIVO	BUFFER KERNEL
1. NESSUN BUFFER 2. SINGOLA RIGA (newline char) 3. BUFFER COMPLETO (max dim)	=> RAGGRUPPA LE SCRITTURE (perchè le letture sono più veloci delle scritture)
Flush > svuotamento buffer Applicativo	Sync > svuotamento buffer Kernel

Comandi utili per l'analisi del sistema

1. Vedere se un hd è SSD o HDD
2. Vedere memoria (RAM)
3. Versione di linux
4. -Altro comando:
5. Spazio disponibile su dischi
6. Rete
7. PCI dvices
8. USB devices
9. BLOCK devices
10. CPU info
11. Tempo di uptime
12. Stats hardware

1. lsblk -o name,rota (1 = HHD 0 = SSD)
2. free
3. cat /proc/version
4. uname -v
5. df -h
6. lfconfig
7. lspci
8. lsusb
9. lsblk
10. lscpu
11. uptime
12. sudo lshw + pass // OPZ: -short

Processi

Astrazione di un programma in esecuzione

Una ASTRAZIONE è una implementazione software:

ASTRATTA
Il SO traduce poi questa astrazione in azioni concrete sull'hardware

SEMPLIFICATA
Deve semplificare la gestione all'utente, quindi nasconde tutti i dettagli suoperflui

Ogni ASTRAZIONE è composta da 2 parti:

Strutture Dati

- Rappresentazione di uno stato interno
- Puntatori a risorse in uso

Funzioni gestione

- Creazione
- Terminazione
- Esecuzione
- Comunicazione
- Monitoraggio

Approccio generale

ACB
Descrittori astrazione
(Abstraction Control Block)

Approccio sui processi

PCB

Descrittori di Processo

(Process Control Block)

Scopi di utilizzo

1. Esecuzione molteplici programmi e molteplici processi
   ⇒ TIME SHARING
   Esecuzione in quanti di tempo (interruptions)

2. ⇒ SICUREZZA

3. Controllo Risorse
   ⇒ MONITORAGGIO

Sicurezza OS

Argomenti

Controllo degli Argomenti di una System Call

Modalità Duale

1. modo **utente**

   - Non può accedere ai registri CPU o fare azioni i/o su periferiche

2. modo **supervisor**

System Call

1. **User mode**

Non può modificare memoria di altre applicazioni o El kernel, no può eseguire istruzioni assembly

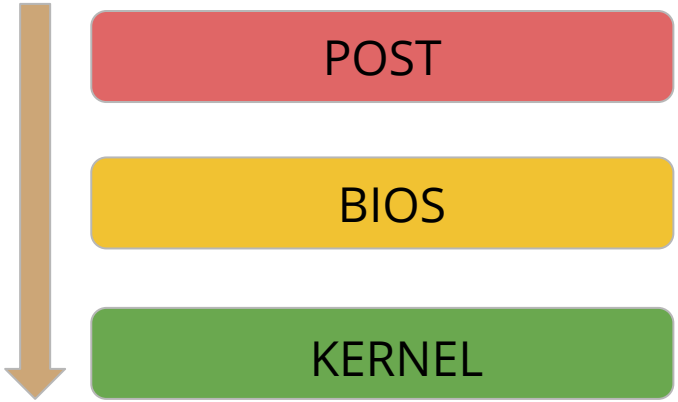
2. **Kernel mode**

===> Privilegi elevati grazie a System Call

Process Tree

Avvio: Da parte di un utente, del SO, da parte di applicazioni un'applicazione può creare altri processi: **FORKING** - padre -> figlio (clone)

PATH: variabile ambiente standard che contiene tutti gli eseguibili



(PowerOn Self Test) test cablaggio, tastiera, RAM, inizializzazione scheda video, dischi e rete

cerca una periferica di avvio (boot loader) e lo carica in memoria. Scelta del SO da caricare

inizializza sbin/init e tutti i servizi

Analisi Dei Processi

Individuare i processi

pidof pidof firefox

pgrep pgrep ‘^testo.*$’

```
adw@adw-HP-Laptop-17-bs0xx:~$ pidof chrome
9964 9950 9260 8016 5988 5975 4216 4211 3508 3501 3474
adw@adw-HP-Laptop-17-bs0xx:~$ pgrep '^chro.*$'
3474
3501
3508
4211
```

ps -p ID

```
adw@adw-HP-Laptop-17-bs0xx:~$ ps -p 9964
  PID TTY          TIME CMD
 9964 tty2     00:00:00 chrome
```

top

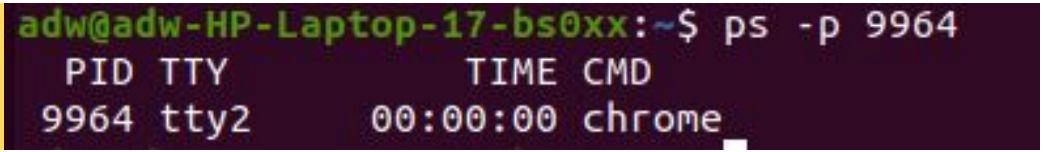
-f (full) -F extra full -l (long)

```
top - 09:12:23 up  2:32,  1 user,  load average: 0,30, 0,49, 0,61
Tasks: 271 total,   1 running, 269 sleeping,   0 stopped,   1 zombie
%Cpu(s):  9,1 us,  2,9 sy,  0,0 ni, 87,7 id,  0,1 wa,  0,0 hi,  0,2 si,  0,0 st
MiB Mem :  15911,0 total,  11725,7 free,   1736,3 used,   2449,0 buff/cache
MiB Swap:   2048,0 total,   2048,0 free,      0,0 used.  13557,0 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2700 adw       20   0 2905756 236236 117636 S  10,7   1,4   4:15.46 gnome-she+
 1540 adw       20   0  972220  77140  53860 S   5,7   0,5   3:53.55 Xorg
 3474 adw       20   0 1026092 263080 158240 S   4,0   1,6   5:31.37 chrome
 9260 adw       20   0  914152 355404  95636 S   3,7   2,2  11:52.48 chrome
 4211 adw       20   0  385712 113396  76312 S   2,0   0,7   5:25.26 chrome
```

ps

ps -p ID



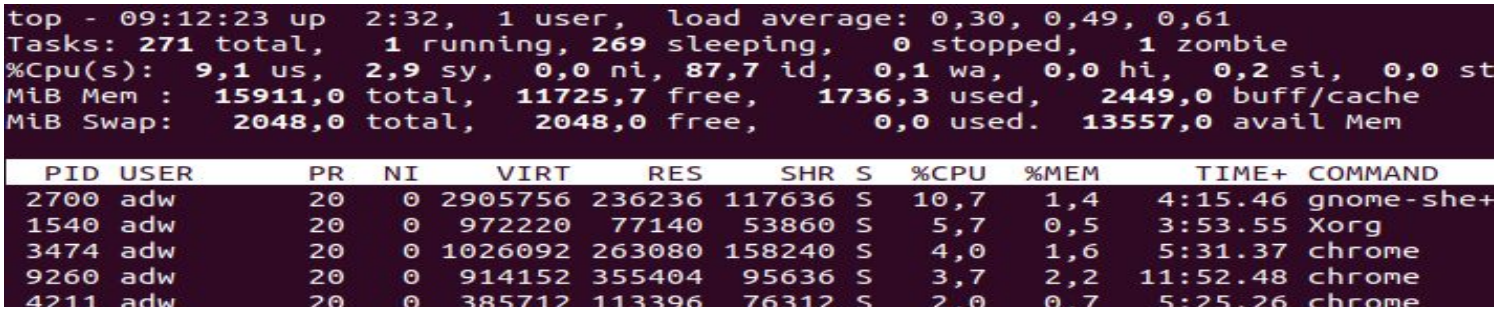
TTY: terminale a cui è agganciato il processo. Può essere:
- **tty** => terminale nativo in ambiente testuale
- **pts** => emulazione del terminale (pseudo terminal) es. SSH
- **vty** => (virtual) es. Router

TIME: tempo di CPU consumato dal processo

CMD: nome del comando

Debian: **CTRL+ALT+F3** per andare in modalita tty / **CTRL+ALT+F2** per tornare indietro
(Ogni distribuzione può avere comandi diversi, fate attenzione)

Top



SOPRA : Indici di prestazione del sistema

Carico medio (load average) è la media esponenziale del numero di processi in stato RUNNING.
Sono calcolate le medie su un intervallo di 1, 5 e 15 minuti.
Se il carico medio è maggiore del numero di CPU ⇒ stallo

Tasks
Utilizzo di CPU è la frazione di tempo in cui la CPU si trova in un determinato stato.

- **us**: esecuzione in modo utente
- **sy**: esecuzione di chiamate di supervisore
- **id**: CPU inattiva

Utilizzo di memoria (fisica e area di swap)

- **total**: memoria a disposizione
- **used**: memoria usata
- **free**: memoria disponibile

TABELLA DEI PROCESSI

PID
USER
PR: priorità del processo.Più è bassa > importanza
RES: memoria principale assegnata al processo
%CPU: somma uso di CPU in modo utente (us) + supervisore (sy)
%MEM: uso memoria principale
%TIME+: tempo di CPU consumato dal processo
COMMAND: nome del comando

Top

COMANDI DI TOP

h: invoca l'help in linea.
s: imposta l'intervallo di campionamento.
f: seleziona il campo di ordinamento.
R: ordinamento crescente/decrescente.
V: visione ad albero.
l: indici di sistema
Cursore su/giù, **Pagina** su/giù: navigazione dell'elenco dei processi.

modalità non interattiva

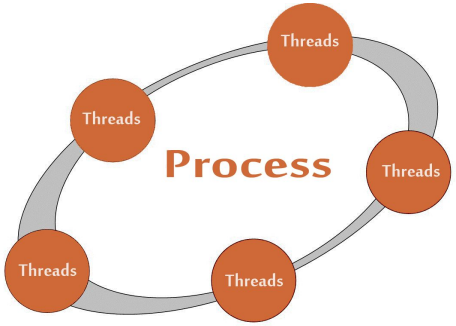
-b:

===> top stampa il suo output su terminale, in modalità batch

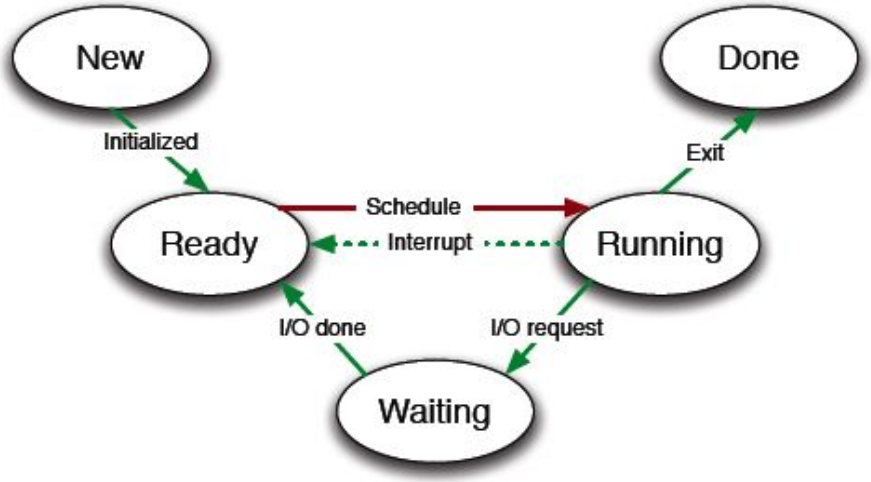
Es: (d 1 = intervallo 1 secondo)
top -b -d 1

Thread

E’ una astrazione di esecuzione di un programma alternativo al processo. Può eseguire codice condiviso col processo che lo ha generato (riduzione di memoria) ma non può caricare applicazioni diverse (svantaggio).



Stati Dei Processi



Terminare i Processi

Kill -l

⇒ Lista di tutti i segnali di interruzione

```
adw@adw-HP-Laptop-17-bs0xx:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

REPARENTING / ZOMBIE:

Quando viene kilato un processo padre, i suoi processi figli diventano “**orfani**” ⇒ processo di **reparenting**
OPPURE
Processi ZOMBIE (architett. moderne)
Status code **(Z)**
= occupa ancora memorie, viene chiesto al processo padre di gestire la chiusura del figlio prima del suo kill

CTRL + C
= SIGINT
Chiusura controllata

CTRL + \
= SIGQUIT
chiusura rapida

Kill PID
= SIGTERM PID
Chiusura controllata

Kill KILL PID
= SIGKILL = kill -9 PID
Chiusura rapida

Bloccare i Processi

Sospensione

CTRL+Z
= kill -STOP PID

⇒ il processo viene sospeso (T)
⇒ manitiene tutte le risorse allocate in memoria
⇒ si mette in attesa di un segnale

jobs

=> per vedere i processi bloccati

```
adw@adw-HP-Laptop-17-bs0xx:~$ jobs
[1]+  Fermato                 top
```

bg (n)
=> sblocca ed esegue in background
(N) è il numero del job, NON del processo

```
adw@adw-HP-Laptop-17-bs0xx:~$ bg 1
[1]+ top &
```

⇒ NOTA: & (alla fine del comando) = eseguire in background

fg (n)
=> sblocca e torna in foreground
(N) è il numero del job, NON del processo

```
adw@adw-HP-Laptop-17-bs0xx:~$ fg 1
```

⇒ torna in foreground

Schedulazione Dei Processi

I processi sono gestiti in base alle **PRIORITA'** ===> il KERNEL attiva diversi **SCHEDULER** (unoo per risorsa)

SCHEDULER

FASI:

1. ACCODAMENTO (CODE)
2. ALGORITMO DI SELEZIONE
3. DISPATCHER (Assegnazione della risorsa a chi la chiede)

Richiesta => Attesa > Liberazione Risorsa > Risveglio > Uso

PRELAZIONE (PREEMPTION)
Il kernel può decidere di riassegnare le priorità di un processo

CODE

CODE DI ATTESA
(WAIT QUEUE)

CODE DI PRONTO
(READY QUEUE)

Sono costituite da:

- Strutture di Controllo
- Funzioni

⇒ ne esistono molte: 1 x ogni evento!

Es. Schedulazione Dei Processi: CPU scheduler

Durante l'esecuzione di un processo, si alternano 2 fasi:

CPU BURST	Elaborazione CPU utente o supervisore)
I/O WAIT	Attesa di un evento

Obiettivi dello Scheduler CPU:

- Mantenere un buon grado di multiprogrammazione
- Mantenere i giusti processi:
  Desktop => **I/O Bound**
  Server => **CPU bound**

Bound = vincolato ad una risorsa o evento

Processi CPU Bound:
Poche burts cpu lunghe / Poche richieste I/O

Processi I/O Bound:
Tante burts cpu brevi / / Molte richieste I/O

Multiprogrammazione =

N° di processi attivi pronti

+

N° processi in esecuzione

Priorità dei processi

=> indicato da un num (int)

Problema da evitare: **STARVATION** (=Processo che muore di fame)

PRIORITY BOOST (AGING)

Lo scheduler individua i processi in starvation
e alza gradualmente la sua priority

LIVELLI DI PRIORITA'
(140)

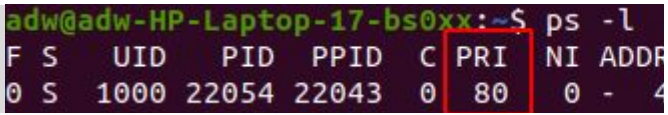
0 - 139

0 + alta
139 + bassa

STATICHE (o normali)
Da **100** a => **139**

Real Time (alta precedenza)
Da 1 a => 99

FIFO / ROUND ROBIN

TOP (PR)	{ker -100} ⇒ 0 / 39	
PS (PRI)	{ker -40} ⇒ -40 / 99	

Nice e Renice

=> modifica di priorità STATICHE

NICENESS	= VALORE DI CORTESIA	[da -20 / a 19]

Valore di cortesia = [Priorità del nucleo - 120]
Il valore di cortesia di default è [0]

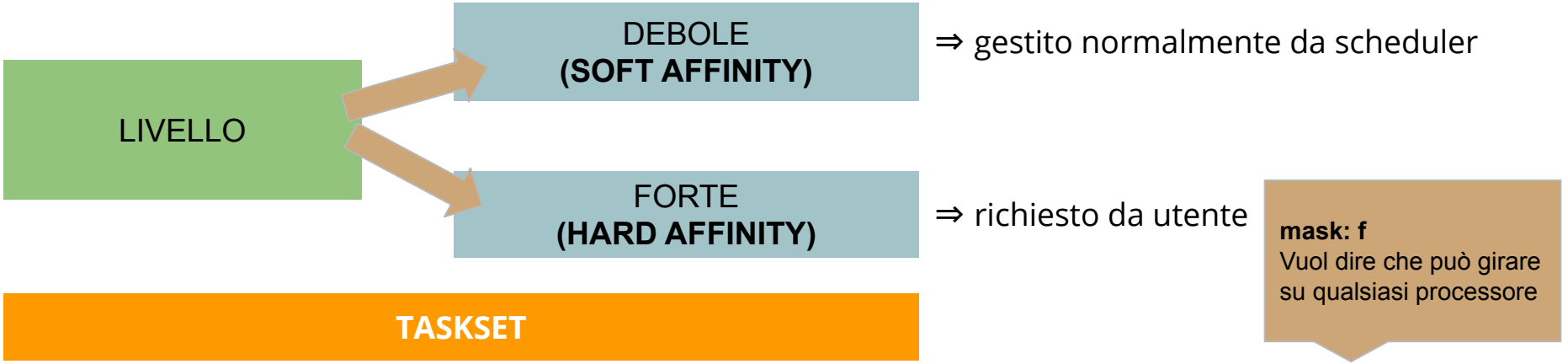
Cortesia perchè più è “cortese” più lascia correre avanti gli altri, metaforicamente.
⇒ quindi più è alto il valore, più il processo avrà bassa priorità.

Valore **[+ 19]** equivale alla statica / normale **[139]**
Valore **[0]** equivale alla statica / normale **[120]**
Valore **[-20]** equivale alla statica / normale **[100]**

NICE	Si applica ad un processo NON ancora eseguito. *Es: nice +15 <comando>*
RENICE	Si applica ad un processo in esecuzione. *Es: renice -13 -p #PID*

AFFINITA' DI CPU (CPU AFFINITY)

= tendenza di un processo a rimanere in esecuzione su una stessa CPU
(Vantaggio: condivisione di risorse!)



stampa	Stampa su quali CPU [0,1,..] gira il processo *Es: taskset -p #PID*
impostazione	Definisce su quali CPU [0,1,..] far girare il processo *Es: taskset -pc 0,1 #PID*

File System

“Struttura organizzativa che regola il modo in cui i files e le directory sono gestiti e manipolati nel SO”

Come è fatto un file

FIle *= sequenza di byte*

- *Memorizzata su un supporto secondario,*
- *IDENTIFICATO da un nome arbitrario*
- *Contenitore di dati arbitrari*

EOF
= end of file

Un file è composto

da 2 parti:

Blocchi di dati

Dati salvati in blocchi all'interno di un supporto (o bock device = dispositivo a blocchi"). Questi blocchi hanno dimensione predeterminata (512 byte)

FBC (File control Block)

Strutture Di controllo

(Contiene proprietà come proprietario, date di creazione, modifica, ultimo accesso, dimensione, permessi di accesso, puntatori a risorse)

Classificazione Files

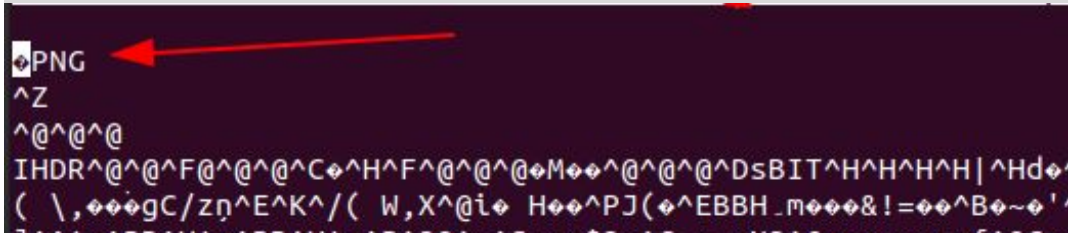
ESTENSIONE

Associazione tramite **estensione** dichiarata (es. .ttx. .pdf)

Liste di **estensioni predefinite** per ciascun **programma**

ANALISI DEL CONTENUTO

Lettura del **MAGIC NUMBER** - Approccio “signature based” Ogni file contiene uno o più caratteri che ne identificano il tipo. Ad esempio nei primi byte.



Contenuto di un file

BINARI → Informazione in linguaggio macchina

TESTUALI → Devono essere associati ad un tipo di **CODIFICA** per essere letti

Esempio:
ASCII 1 byte = 1 carattere
UTF8 (Dimensione variabile da 1 a 4 byte)

https://it.wikipedia.org/wiki/ASCII#ASCII_ed_UTF-8

ESEGUIBILI

I files eseguibili sono associati ad un **caricatore** (**LOADER**) che associa il file ad un **interprete**. Carica quindi le **librerie** necessarie per la sua "esecuzione" da parte di questo interprete.
(Es: #!/bin/bash è un interprete o #!/usr/bin/python)

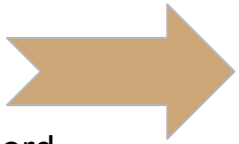
Organizzazione interna

Fisica

- I dati sono impacchettati in blocchi del disco

Logica

- Flusso di byte ⇒ sequenza
- Record logici ⇒ 1 riga = 1 record
- Con Indice ⇒ Database



Accesso Ai FIles

- A “Nastro” ⇒ sequenziale
- Accesso Diretto
- Con Indice

Tabella dei files aperti

Ogni PCB contiene un puntatore ad una tabella di files aperti dal processo.

⇒>Comando **lsof**
lsof -p #PID
lsof -u <username>
Lsof -t <file-name>

Indice Doppio ⇒ 2 file

- 1. <key, value>
- 2. record (value)

Indice Gerarchico: (per database di grandi dimensioni)
Schema gerarchico a più livelli:

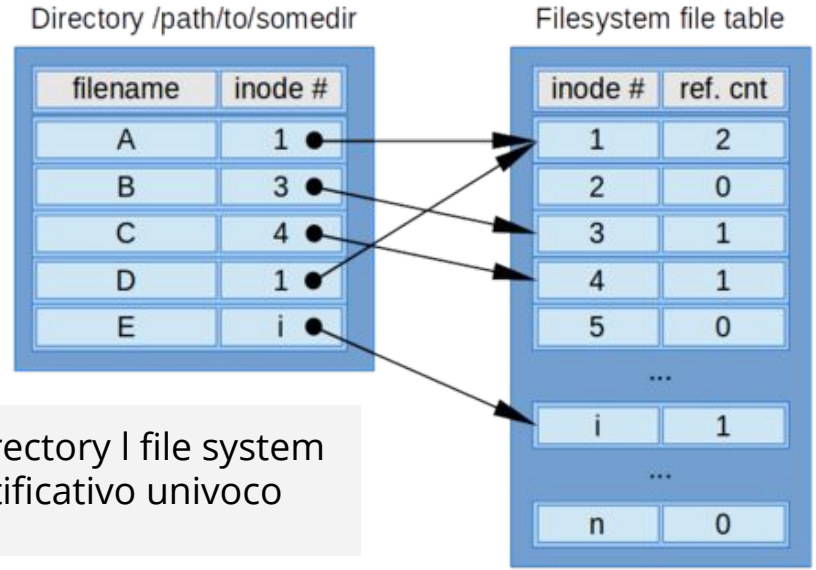
- Indice primario (puta al secondario)
- Indice secondario (punta al record)

Directory

= file contenente un nome + puntatore (ad una struttura con metadati)

Strutture di directory

- Singolo Livello
- Due livelli
- Ad Albero
- A grafo aciclico
- **A grafo ciclico** (possibili cicli infiniti)

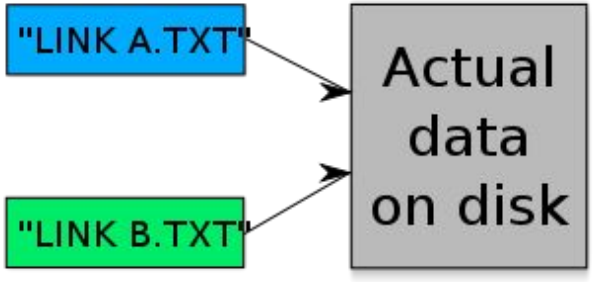


inode

Ad ogni file e directory l file system associa un identificativo univoco (inode)

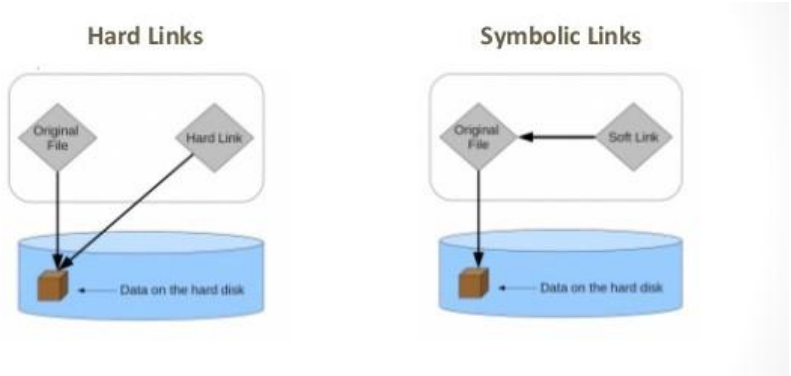
Hard Link

- E’ l’associazione di un nome ad un contenuto (inode)
- il collegamento fisico deve trovarsi sullo stesso file system che contiene i dati a cui fa riferimento
- un collegamento fisico non può restare “orfano”



Soft Link (SIMBOLIK o SYMLINK)

- Può chiudere un “ciclo” infinito.
- Metadati e contenuto: sono unici
- E’ un link simbolico (ad un hard link)
- Possono essere creati fra molteplici Files System (se non presente ⇒ rosso)
- Possono essere creati fra directory superiori



Cancellazione files

Link Counter: E’ un conteggio di quante volte un file è referenziato. Quando un file viene “cancellato”, non viene eliminato del tutto, ma solo scollegato ed il conteggio settato a zero.

Montaggio e smontaggio

ROOT FILE SYSTEM

Almeno 1 File System deve essere presente al **momento dell'avvio** (es /sbin o /usr/bin) per avviare i servizi.

MOUNT POINT

Un file System secondario per essere utilizzato deve avere associato ad un dispositivo di memoria secondaria su un punto di "montaggio" (una **directory**) che li collega.

OPZIONI DI MONTAGGIO

E' possibile montare un file System in diverse modalità:

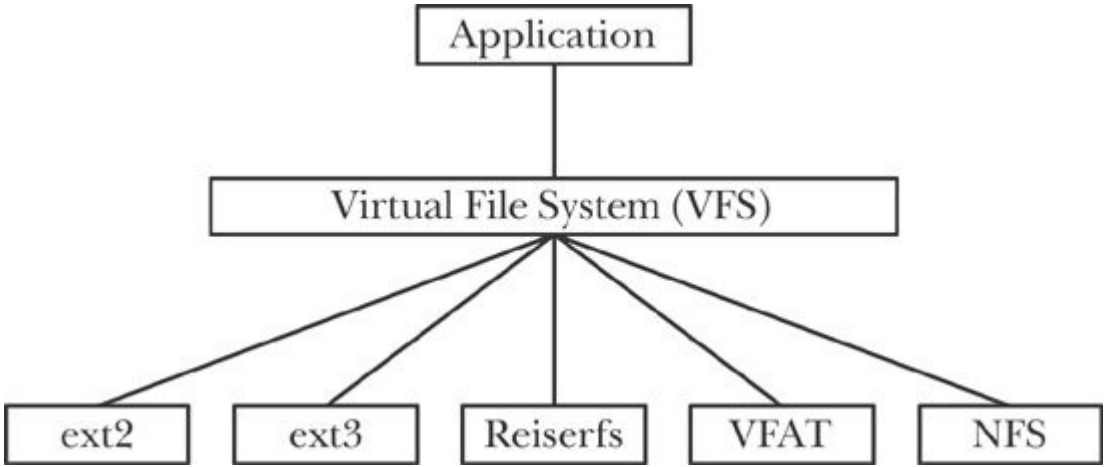
- Read only
- Exec (permette esecuzione di programmi)
- Sync (scrittura sequenziale)
- Async (scrittura asincrona - in blocco)

UNMOUNT

Un file System secondario può essere scollegato con una operazione di **smontaggio.** Questa operazione è preceduta da un **FLUSH dei buffer del kernel**. Poichè l'operazione su un file system sono eseguite in blocco (per motivi di efficienza) estrarre un dispositivo esterno senza prima averlo smontato può causare la perdita dei dati.

Virtual File System

E’ una astrazione software che si interpone (interfaccia) fra il File System fisico ed il kernel, per superare varie limitazioni ed essere indipendente dai dispositivi. Permette quindi un collegamento fra periferiche hardware secondarie esterne e locali ed il kernel. Il VFS decompone il percorso di un file ed individua il percorso del dispositivo su cui effettivamente si trova (path lookup).



Bash

esempi

Navigazione terminale 1/2

1. TAB
2. CTRL + a
3. CTRL + e
4. CTRL + b
5. CTRL + f
6. CTRL +xx
7. CTRL + w
8. CTRL + u
9. CTRL + k
10. CTRL + y
11. CTRL + CTRL -
12. CTRL + C
13. CTRL + L
14. CTRL + D

1. Autocompletamento
2. Sposta il cursore all'inizio
3. Sposta il cursore alla fine (end)
4. Indietro di 1 carattere (back)
5. Avanti di 1 carattere (forward)
6. Avanti e indietro fra inizio e fine
7. Cancella ultima parola
8. Cancella tutto prima del cursore
9. Cancella tutto Dopo del cursore
10. PASTE di quanto tagliato (con w / u / k)
11. Aumenta / riduce caratteri
12. BREAK di un processo
13. CLEAR del terminale
14. Logout (exit)

Navigazione terminale 2/2

1. CTRL + p (o freccia in su)
2. CTRL + n (o freccia in giù)
3. CTRL + r
4. ^parolavecchia^parolanuova
5. CTRL + d
6. ALT + d
7. ALT + c
8. ALT + l
9. ALT + u

1. Comando precedente (previous)
2. Comando successivo (next) - dopo p
3. Ricerca fra la history dei comandi
4. Sostituzione voce ultimo comando
5. Cancella il carattere sotto il cursore
6. Cancella il carattere dopo il cursore
7. Capitalize carattere sotto cursore
8. Lowercase carattere sotto cursore
9. Capitalize caratteri dal cursore alla fine

BASH

1.	&	1. Esegue in Background
2.	&&	2. Esegue solo se il comando precedente va a buon fine
3.	||	3. Oppure (nel caso il comando precedente non vada a buon fine)
4.	>	4. Redirection
5.	<	5. Input (es da file)
6.	>>	6. Redirection append mode
7.	2>	7. Standard error redirection
8.	1>	8. Standard Output redirection
9.	2>&1	9. Redirect dello STND err verso STND Ouput
10.	1> & 2	10. Redirect dello STDN Otpt verso STND error

GLOBBING, Comandi, Argomenti etc

?	nessun carattere o 1 carattere
*****	nessun carattere o qualsiasi numero di c.
[ ]	match ES [1-5] [esd]
!	Esclusione
[^	Esclusione
^	Inizia con
^ $	Fine sequenza ^ - $
exit	Valore di ritorno

[ ]	Condizione fra parentesi if [ ]
[[ ]]	\\ senza double quote “ “
$(( ))	Direttive matematiche N=$((N+5))
seq n-n	Cicli es seq 10 // seq 5 10
$1,$2, ..$n	Argomenti
$0	Nome di invocazione
$#	Numero argomenti
|	Pipeling
export	Crea variabili di ambiente

Eseguire uno script

Se non eseguibile:

bash nomefile

Se eseguibile:

./nomefile

Dare i permessi esecuzione: **chmod +x** nomefile

Esempio di condizionale

```
if              [[ <condizione> ]]
then
                <comandi>
elif    [[ <condizione >> ]]
then
else
                <comandi>
fi
```

Esempio ciclo for

```
for V in $(seq 5 10);
do
echo $V
done
```

Esempio ciclo while

```
while [[ $N -lt 10 ]]
do
N=$((N + 1))
echo $N
done
```