GNU/Linux Intermedio

Terza lezione:

AVVIO DEL SISTEMA OPERATIVO

Il processo di avvio

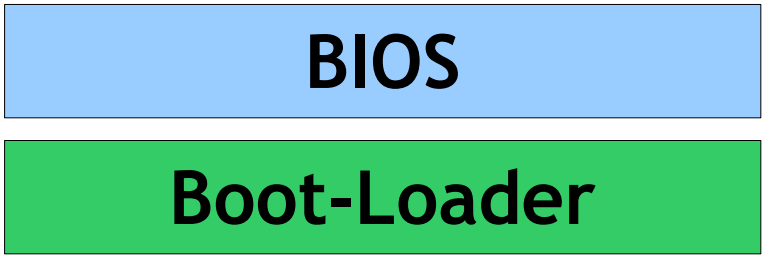
- Abbiamo già accennato un paio di volte al meccanismo di avvio di un sistema GNU/Linux.
  - Abbiamo parlato del boot-loader e della sua configurazione
  - Abbiamo parlato del funzionamento del kernel
  - Abbiamo accennato al file `/etc/fstab`
- E' giunto il momento di approfondire questo aspetto, e vedere nel dettaglio il meccanismo di avvio di Linux.
- Vedremo anche quali problematiche sono legate alle prime fasi dell'avvio
- Vedremo funzionamento, organizzazione e configurazione degli script di avvio (in sistemi System V e non)
- Vedremo infine come vengono avviati (ed in seguito gestiti) i cosi detti “demoni”, tramite gli script di avvio

Step 1: il BIOS

- Nel momento in cui al computer viene data l'alimentazione, il controllo dell'esecuzione è in mano al BIOS, che effettua una procedura di controllo hardware chiamata POST (Power On Self Test)
- Nella configurazione del BIOS è previsto un “ordine di boot”.

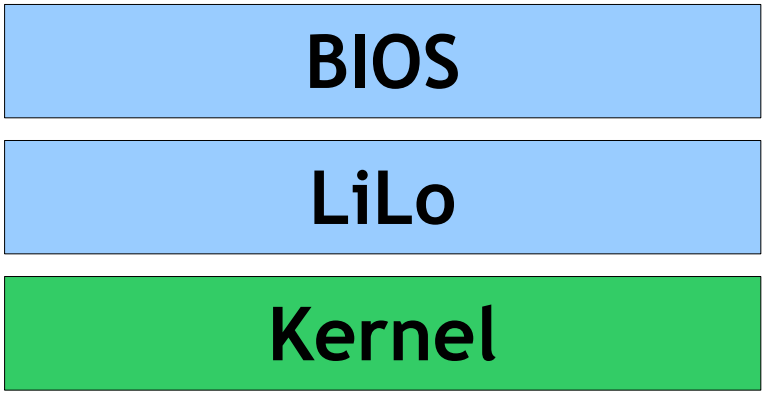
Step 2: l'MBR

- Terminato il POST, il BIOS legge i primi 512 bytes del disco di avvio (che prendono il nome di MBR) e li utilizza come “boot-loader”, passandogli il controllo.
- Nell'MBR può esserci un boot-loader intero (es. LiLo) o una parte di esso (es. Grub)



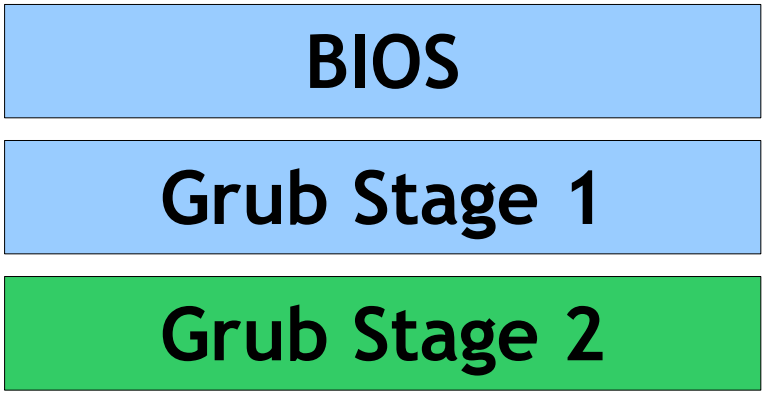
Step 3: LiLo

- Terminato il POST, il BIOS legge i primi 512 bytes del disco di avvio (che prendono il nome di MBR) e li utilizza come “boot-loader”, passandogli il controllo.
- Nell'MBR può esserci un boot-loader intero (es. LiLo) o una parte di esso (es. Grub)
- LiLo conosce già l'indirizzo fisico del kernel, che provvede a caricare ed eseguire.



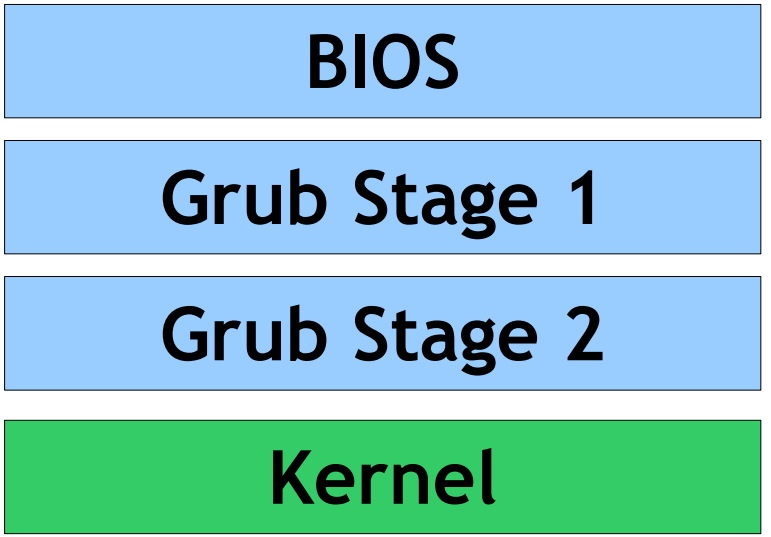
Step 3: Grub

- Il boot-loader Grub invece, è piu complesso. Nell'MBR risiede un primo “stage” di Grub, che si occupa di caricarne un secondo residente su disco.
- Questo primo stage contiene le informazioni necessarie ad accedere il filesystem su cui risiede il secondo stage (che può essere di 9 diversi tipi, tra i quali Fat32, ReiserFS, ext2 o 3, NTFS e JFS.



Step 4: Il kernel

- Il secondo stage invece, si occupa in vivo di mostrare a schermo il menu di avvio, individuare l'indirizzo fisico del kernel da caricare, caricarlo in ram e passargli il controllo dell'esecuzione.
- Una volta che il controllo è in mano al kernel, indipendentemente da quale boot-loader lo ha lanciato, il kernel si occupa di portare a termine l'avvio del sistema



Step 5: il filesystem /

- Per prima cosa è necessario individuare la partizione di root. Tutti i files di configurazione, compresi quelli preposti alla gestione del processo di avvio, risiedono infatti su questa partizione, ed in particolare all'interno della directory `/etc`
- Il nome della device che contiene il filesystem `/` viene specificato direttamente dal boot-loader, come abbiamo già visto osservandone i files di configurazione.
- Nel caso in cui questa partizione non esista, il kernel si troverà nell'impossibilità di proseguire e il processo di avvio si arrenerà di fronte a questo scoglio insormontabile.
- Tutto quello che si ottiene è un bel “kernel-panic”.

Step 6: init (?)

- In caso contrario, il filesystem verrà montato (*) come indicato dal boot-loader:
  - In lettura/scrittura
  - In sola lettura (rimontato rw dopo fsck)
- Nel secondo caso (la prassi), verrà eseguito un controllo di integrità sul filesystem prima di rimontarlo in lettura/scrittura e proseguire con il processo di avvio, caricando tutti i moduli necessari (o configurati) e infine avviando il processo “init” che si occuperà (vedremo) di gestire il resto del processo d'avvio.

- (*) Vi è però un problema. Al fine di poter montare il filesystem di root, il kernel ne deve conoscere il tipo ed i dettagli implementativi. Queste informazioni dove sono?

Un gatto che si morde la coda

- Nel modulo del kernel preposto alla gestione di quel filesystem.
- Distinguiamo allora 2 casi:
  - Il modulo è “compilato staticamente” all'interno del kernel stesso
    - Nel qual caso, il kernel conosce già di per se tutti i dettagli necessari a montare quel tipo di filesystem e quindi potrà banalmente proseguire indisturbato.
  - Il modulo va caricato (è esterno) e si trova sul filesystem di root, all'interno della consueta directory `/lib/modules/2.6.X/*`, filesystem del quale non sono noti i dettagli implementativi.
    - E' un gatto che cerca di mordersi la coda. Come risolvere il problema?

Il ram-disk initrd

- Nasce l'idea del “ram-disk” initrd (initial ramdisk).
- Si tratta di un file (anche compresso), che contiene un filesystem minimale, contenente tutti i moduli necessari a provvedere al montaggio della `/` (ma non solo: consente ad esempio di decifrare una disco cifrato)
- Questo file viene specificato in fase di configurazione del boot-loader, e caricato in RAM dal kernel all'avvio.
- Il kernel, utilizzando questo filesystem minimale come root temporanea, carica il modulo necessario a “decodificare” le informazioni contenute sul vero filesystem `/`
- Il modulo viene quindi utilizzato per montare il filesystem `/` in modo da sostituire il filesystem ramdisk, e proseguire il processo di avvio, giungendo ad eseguire “`/sbin/init`”

Step 6: init (!)

- “init” è il primo processo lanciato sul sistema (ha infatto PID [process id] numero 1) ed ha il compito di “generare processi” a partire da un file di configurazione, che si trova nella cartella `/etc`, e si chiama “inittab”
- Ad una prima occhiata, la sintassi del file `/etc/inittab` potrebbe sembrare piuttosto complessa, in realtà è piu semplice di quanto non si pensi.
  - Ogni riga rappresenta un'operazione da eseguire, le condizioni e le modalità di esecuzione di questa operazione
  - La sintassi di ogni riga contiene diversi campi, separati dal carattere “:”, secondo la struttura

```
[ID]:[RUNLEVELS]:[ACTION]:[PROCESS]
```

I runlevels

- Il primo campo, “id”, è un identificativo univoco all'interno del file, composto da al piu 4 caratteri (solitamente 2)
- Il secondo campo invece riguarda il/i “runlevel” per i quali quella data operazione deve essere eseguita.
- L'avvio di Linux infatti, può essere differenziato in piu “livelli”, solitamente numerati da 0 a 6, per ognuno dei quali possono essere specificate differenti configurazioni di avvio.
- Solitamente i runlevels vengono “personalizzati” già dal mantainer della distribuzione, che ne indica poi le caratteristiche come commenti nelle primissime righe di inittab

I runlevels

```
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
```

- Questi sono i runlevel descritti per la distribuzione “Debian” nel file `/etc/inittab`
- Debian, a differenza di molte altre distribuzioni, ha preferito recentemente non differenziare i runlevels da 2 a 5, lasciando all'utente l'onere della relativa configurazione.
- Il runlevel 0 è quasi sempre dedicato allo spegnimento del sistema, mentre il numero 6 al suo riavvio (spesso si tratta di un unico runlevel che si distingue solo al termine dello spegnimento, a seconda che si debba riavviare o no)

I runlevels

```
#  Runlevels:
#    0       Halt
#    1(S)    Single-user
#    2       Not used
#    3       Multi-user
#    4       Not used
#    5       X11
#    6       Reboot
```

- Il runlevel 1 è quasi sempre dedicato alla modalità “mono-utente” utilizzata piu che altro per quelle operazioni di manutenzione in cui è necessario che nessun altro utente sia collegato al sistema.
- Gli altri runlevel, su molte distribuzioni, sono differenziati a seconda che configurino o meno la rete dati, che avviino o meno l'interfaccia grafica, e cosi via (nell'esempio, i runlevel della distribuzione ArchLinux)

Gli script di avvio

- Ogni singola operazione che compone un runlevel, viene gestita eseguendo uno “script di avvio”
- Tutti gli script di avvio, si trovano solitamente in una delle seguenti cartelle (a seconda della distribuzione):
  - `/etc/init.d/`
  - `/etc/rc.d/`
- Avremo modo di parlare approfonditamente di questi script, perchè non vengono utilizzati solo per la gestione del processo di avvio.
- Esistono due diverse filosofie di organizzazione degli script di avvio:
  - System V
  - BSD

L'avvio “BSD”

- Nella filosofia “BSD” (tra le distro piu note, utilizzano questa filosofia praticamente solo da Slackware e da ArchLinux), tutto l'avvio di ogni singolo runlevel è specificato in un unico script, che esegue tutte le operazioni necessarie (rc1, rc2, rc3...)
- Visto che si tratta della filosofia meno diffusa, spesso si finisce con l'introdurre soluzioni “ibride”, con lo script “unico” che viene suddiviso in piu script (uno per operazione) che vengono però chiamati secondo una concatenazione particolarmente rigida.
- Secondo i detrattori di questa filosofia, se ne perde in prestazioni e flessibilità, mentre secondo i detrattori della filosofia “System V”, questa presenta una complessità eccessiva.

L'avvio “System V”

- Nella filosofia “System V” (la piu diffusa, che prende il nome dal ramo di Unix che la implementava), ogni operazione è gestita da un singolo script.
- Esiste una cartella per ogni runlevel, solitamente con il nome di:
  - `/etc/rc.`***N***`/`
  - `/etc/rc`***N***`.d/`

(dove ***N*** è il numero del runlevel), che contiene un link simbolico ad ognuno degli script che devono essere eseguiti per quel determinato runlevel. Il nome del link comincia con S***X***- (***X*** num., ordinale) se è deve “avviare” il processo, o con K***X***- se deve “arrestare” il processo.

- L'esecuzione di ognuno di questi script è affidata ad uno script “generale”, eseguito da init e specificato in inittab

L'avvio “Upstart”

- “Upstart” è un sostituto di init, inizialmente introdotto da Ubuntu (Edgy Eft, 6.10)
- E' un demone, che legge `/etc/event.d/` e ne gestice il contenuto: un file, un'azione
- Pienamente retrocompatibile con SysVInit (questo ne ha determinato il successo)
- Vantaggi:
  - Velocità
  - Parallelizzazione (e gestione dipendenze)
  - Rinnovamento
  - Handling semplificato (initctl, status, start, stop)
- 0.3.9 – pecca ancora di instabilità in termini di config.

/etc/inittab: initdefault

- La prima riga del file (a parte i commenti, che sono preceduti dal carattere # come nella maggior parte dei files di configurazione e degli scripts) assomiglia quasi sempre a:

```
id:5:initdefault:
```

- L'opzione initdefault serve a definire il runlevel di default, in questo caso “5”. Si tratta di una riga “anomala”, in quanto è l'unica in cui il numero del runlevel non serve a specificare per quali runlevel eseguire questa operazione.
- Per cambiare il runlevel di default del sistema, è sufficiente sostituire in questa riga il valore (nell'esempio 5) con quello del runlevel che si desidera venga avviato.
- Per cambiare runleve invece, basta digitare “init ***N***”

/etc/inittab: i runlevels

- Come dicevamo, la riga “id:” è una riga anomala. Per ogni altra riga del file inittab infatti, i numeri specificati nel campo `[RUNLEVEL]` definiscono per quali di questi quell'operazione deve essere eseguita.
- Ad esempio
  - `[ID]:234:[ACTION]:[PROGRAMMA]`
    - `[PROGRAMMA]` viene eseguito per i runlevel 2, 3 e 4
    - `[PROGRAMMA]` non viene eseguito per 0, 1, 5 e 6
  - `[ID]:123456:[ACTION]:[PROGRAMMA]`
    - `[PROGRAMMA]` viene eseguito per tutti i runlevel
  - `[ID]:4:[ACTION]:[PROGRAMMA]`
    - `[PROGRAMMA]` viene eseguito per il solo runlevel 4.

/etc/inittab: le actions

- Come possiamo facilmente intuire, l'ultimo campo (`[PROGRAMMA]`) riporta il comando (o il nome dello script) che deve essere eseguito
- Il campo `[ACTION]` invece, definisce la tipologia dell'opzione. Questa può essere una configurazione di init (ad esempio abbiamo già visto `initdefault`) oppure una modalità di esecuzione del comando specificato:
  - `respawn`
    - Fa si che all'uscita del processo lanciato, questo venga riavviato. E' utile in particolare per le consoles: viene lanciato il comando “getty” (una console) e quando questa termina (l'utente esegue il logout) viene automaticamente rilanciata e torna in attesa di un nuovo login

/etc/inittab: le actions

- `wait`
  - Fa si che il processo di avvio attenda il termine del comando lanciato da questo record prima di proseguire.
  - Utile per tutte quelle situazioni in cui è indispensabile specificare rigidamente l'ordine con cui devono essere eseguiti determinati comandi
- `ctroaltdel`
  - Definisce come comportarsi a fronte di un Ctrl+Alt+Del
- `sysinit`
  - Con questa action, si chiede di ignorare il campo “runlevel” ed eseguire questo comando in qualsiasi caso all'avvio del sistema.
- `powerwait, powerfail, powerokwait, powerfailnow`
  - Consentono di gestire tutti quegli eventi che riguardano l'alimentazione del sistema (molto utili per i portatili)

/etc/inittab: gli script di avvio

- Nel caso piu comune, il file inittab inizia con la riga “id:”, al quale segue la definizione degli script da eseguire per i diversi runlevel, ed infine sono inserite le righe di avvio (respawn) dei diversi terminali.
- Di seguito sono riportate queste le righe relative all'inittab di Debian. Viene lanciato “rc” con il numero del runlevel come parametro. “rc” si occuperà di lanciare i diversi script di avvio.

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

Gli script di avvio

- Come dicevamo all'inizio, nel sistemi “System V” (ma alla fine funziona allo stesso modo nei sistemi “BSD”), ogni operazione da eseguire all'avvio è gestita da uno script a se stante.
- Le operazioni che possono caratterizzare un runlevel sono fondamentalmente di 2 tipologie:
  - Configurazione (ed esempio configurazione della rete)
  - Avvio di un servizio/applicazione: spesso e volentieri un “demone”
- Un demone è un applicativo che non comunica direttamente con l'utente del sistema, ma con altri processi, ad esempio via rete.
- Esempio? Il server di posta si avvale di un demone, cosi come il server web, e via dicendo.

Gli script di avvio

- Ogni demone (servizio) ha quindi un proprio file di gestione, contenuto nella cartella `/etc/init.d/` (o `/etc/rc.d/` a seconda della distribuzione) che consente di gestirlo:

`/etc/init.d/postfix start`

  - Avvia il demone di postfix, server di posta

`/etc/init.d/apache stop`

  - Arresta il demone di apache, server web

`/etc/init.d/mysqld restart`

  - Riavvia il demone di mysql, database

- Spesso questi script vengono utilizzati anche per gestire la configurazione del sistema

`/etc/init.d/networking restart`

Simulazione di avvio

- Accensione

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto
- Il boot-loader (LiLo) carica in memoria il kernel (di cui conosce l'indirizzo fisico) e lo esegue, passandogli come parametro l'indirizzo del file initrd

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto
- Il primo stage del boot-loader (Grub) carica il secondo stage (residente sul filesystem)

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto
- Il primo stage del boot-loader (Grub) carica il secondo stage (residente sul filesystem)
- Il secondo stage del boot-loader (Grub) carica in memoria il kernel (di cui ricava l'indirizzo fisico) e lo esegue, passandogli come parametro l'indirizzo del file initrd

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto
- Il primo stage del boot-loader (Grub) carica il secondo stage (residente sul filesystem)
- Il secondo stage del boot-loader (Grub) carica in memoria il kernel (di cui ricava l'indirizzo fisico) e lo esegue, passandogli come parametro l'indirizzo del file initrd
- Il kernel carica in ram il file initrd e lo usa come / temporanea

Simulazione di avvio

- Accensione
- Il BIOS esegue il Power On Self Test
- Il BIOS carica in ram l'MBR (primi 512 byte del disco) e ne esegue il contenuto
- Il primo stage del boot-loader (Grub) carica il secondo stage (residente sul filesystem)
- Il secondo stage del boot-loader (Grub) carica in memoria il kernel (di cui ricava l'indirizzo fisico) e lo esegue, passandogli come parametro l'indirizzo del file initrd
- Il kernel carica in ram il file initrd e lo usa come / temporanea
- Il kernel carica il modulo per gestire il vero /

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /
- Il kernel rimonta in lettura/scrittura il filesystem /

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem `/`, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem `/`
- Il kernel rimonta in lettura/scrittura il filesystem `/`
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /
- Il kernel rimonta in lettura/scrittura il filesystem /
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /
- Il kernel rimonta in lettura/scrittura il filesystem /
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)
- Init legge il proprio file di configurazione:
  - `id:5:initdefault:`

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem `/`, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem `/`
- Il kernel rimonta in lettura/scrittura il filesystem `/`
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)
- Init legge il proprio file di configurazione:
  - `id:5:initdefault:`
    - Il runlevel da eseguire (se non specificato diversamente) è il numero 5.

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /
- Il kernel rimonta in lettura/scrittura il filesystem /
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)
- Init legge il proprio file di configurazione:
  - `l0:0:wait:/etc/init.d/rc 0`
    - Si riferisce al solo runlevel 0. Saltato.

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem `/`, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem `/`
- Il kernel rimonta in lettura/scrittura il filesystem `/`
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)
- Init legge il proprio file di configurazione:
  - `l{1-4}:{1-4}:wait:/etc/init.d/rc {1-4}`
    - Si riferiscono ai runlevel {1-4}, quindi vengono saltati.

Simulazione di avvio

- Il kernel monta in sola lettura il vero filesystem /, andando a sostituire quello del ram-disk
- Il kernel esegue un controllo d'integrità del filesystem /
- Il kernel rimonta in lettura/scrittura il filesystem /
- Il kernel carica tutti i moduli che servono a gestire l'hardware del sistema, e quelli eventualmente specificati in fase di configurazione (di solito `/etc/modules.conf`)
- Il kernel esegue “/sbin/init” (o altrimenti quello specificato dal boot-loader)
- Init legge il proprio file di configurazione:
  - `l5:5:wait:/etc/init.d/rc 5`
    - Esegue “/etc/init.d/rc 5” ed attende la sua uscita prima di continuare con il boot.

Simulazione di avvio

- Lo script “rc” va a prende la directory `/etc/rc5.d/` e legge i nomi dei files ivi contenuti

Simulazione di avvio

- Lo script “rc” va a prende la directory `/etc/rc5.d/` e legge i nomi dei files ivi contenuti
  - I files che cominciano con S vengono lanciati come

    ```
    /etc/rc5.d/{nomefile} start
    ```
  - I files che cominciano con K vengono lanciati come

    ```
    /etc/rc5.d/{nomefile} stop
    ```

Simulazione di avvio

- Lo script “rc” va a prende la directory `/etc/rc5.d/` e legge i nomi dei files ivi contenuti
  - I files che cominciano con S vengono lanciati come

    `/etc/rc5.d/{nomefile} start`
  - I files che cominciano con K vengono lanciati come

    `/etc/rc5.d/{nomefile} stop`
- Viene ad esempio configurata la scheda di rete, vengono avviati alcuni demoni, viene lanciata l'interfaccia grafica.

Simulazione di avvio

- Lo script “rc” va a prende la directory `/etc/rc5.d/` e legge i nomi dei files ivi contenuti
  - I files che cominciano con S vengono lanciati come

    `/etc/rc5.d/{nomefile} start`
  - I files che cominciano con K vengono lanciati come

    `/etc/rc5.d/{nomefile} stop`
- Viene ad esempio configurata la scheda di rete, vengono avviati alcuni demoni, viene lanciata l'interfaccia grafica.
- “rc” termina la sua esecuzione, init prosegue nella lettura dell'inittab

Simulazione di avvio

- Lo script “rc” va a prende la directory `/etc/rc5.d/` e legge i nomi dei files ivi contenuti
  - I files che cominciano con S vengono lanciati come

    `/etc/rc5.d/{nomefile} start`
  - I files che cominciano con K vengono lanciati come

    `/etc/rc5.d/{nomefile} stop`
- Viene ad esempio configurata la scheda di rete, vengono avviati alcuni demoni, viene lanciata l'interfaccia grafica.
- “rc” termina la sua esecuzione, init prosegue nella lettura dell'inittab
  - `c{1-6}:2345:respawn:/sbin/agetty 38400 vc/{1-6} linux`
    - Apre una nuova console sul sul terminale virtuale {1-6}.
    - Nel momento in cui questa termina, viene riavviata.

Avvio terminato

- A questo punto la fase di avvio è terminata. Tutti i servizi che è stato specificato che debbano essere attivati sono attivi, il sistema è configurato e presenta la schermata di login all'utente.
- Questa schermata può essere testuale (una getty/agetty) o grafica (xdm/gdm/kdm) a seconda che si sia impostato l'avvio automatico dell'interfaccia grafica o meno.
- Supporremo per il momento che il sistema non sia stato configurato per avviare X11 all'avvio.
- All'utente viene quindi proposto qualcosa come:

```
Arch Linux 0.8 (Voodoo)   (ganesh) (vc/1)

ganesh login:
```

Avvio terminato

- A questo punto la fase di avvio è terminata. Tutti i servizi che è stato specificato che debbano essere attivati sono attivi, il sistema è configurato e presenta la schermata di login all'utente.
- Questa schermata può essere testuale (una getty/agetty) o grafica (xdm/gdm/kdm) a seconda che si sia impostato l'avvio automatico dell'interfaccia grafica o meno.
- Supporremo per il momento che il sistema non sia stato configurato per avviare X11 all'avvio.
- All'utente viene quindi proposto qualcosa come:

```
Arch Linux 0.8 (Voodoo)   (ganesh) (vc/1)

ganesh login:
```

Avvio terminato

- A questo punto la fase di avvio è terminata. Tutti i servizi che è stato specificato che debbano essere attivati sono attivi, il sistema è configurato e presenta la schermata di login all'utente.
- Questa schermata può essere testuale (una getty/agetty) o grafica (xdm/gdm/kdm) a seconda che si sia impostato l'avvio automatico dell'interfaccia grafica o meno.
- Supporremo per il momento che il sistema non sia stato configurato per avviare X11 all'avvio.
- All'utente viene quindi proposto qualcosa come:

```
Arch Linux 0.8 (Voodoo)   (ganesh) (vc/1)

ganesh login:
```

Avvio terminato

- A questo punto la fase di avvio è terminata. Tutti i servizi che è stato specificato che debbano essere attivati sono attivi, il sistema è configurato e presenta la schermata di login all'utente.
- Questa schermata può essere testuale (una getty/agetty) o grafica (xdm/gdm/kdm) a seconda che si sia impostato l'avvio automatico dell'interfaccia grafica o meno.
- Supporremo per il momento che il sistema non sia stato configurato per avviare X11 all'avvio.
- All'utente viene quindi proposto qualcosa come:

```
Arch Linux 0.8 (Voodoo)   (ganesh) (vc/1)
ganesh login:
```

Avvio terminato

- A questo punto la fase di avvio è terminata. Tutti i servizi che è stato specificato che debbano essere attivati sono attivi, il sistema è configurato e presenta la schermata di login all'utente.
- Questa schermata può essere testuale (una getty/agetty) o grafica (xdm/gdm/kdm) a seconda che si sia impostato l'avvio automatico dell'interfaccia grafica o meno.
- Supporremo per il momento che il sistema non sia stato configurato per avviare X11 all'avvio.
- All'utente viene quindi proposto qualcosa come:

```
Arch Linux 0.8 (Voodoo)   (ganesh) (vc/1)
ganesh login:
```

Il login

- A questo punto, l'utente deve autenticarsi, inserendo username a password.
- La lunghezza della password era limitata a 8 caratteri fino ad alcuni anni fa. Con l'avvendo delle “shadow passwords” (ne parleremo piu approfonditamente), grazie al fatto che viene memorizzato un hash e non la password vera e propria, e che l'hash ha una lunghezza fissa a fronte di qualsiasi lunghezza di input, le password non hanno piu questo genere di restrizione nelle moderne distribuzioni.
- Una volta che l'utente ha inserito username e password corretti, il programma che gestisce la console (getty/agetty) lancia il programma che gestisce la fase di login: “login”

La Bash

- Login si occupa di andare a leggere il file `/etc/passwd` (di cui parleremo) in cui sono salvate tutte le informazioni riguardanti l'utente, tra le quali, la sua “login shell”, che provvede a lanciare, passandogli tutti i parametri necessari.
- Per gli utenti comuni, la login shell piu usata è certamente la Bash (Bourne Again Shell)
- Può essere lanciata in due modalità differenti
  - Modalità “login”
  - Modalità “interattiva”

(anche se possono esistere shell di login interattive)

- A seconda della modalità con cui è stata lanciata, vengono valutati ed eseguiti script differenti. Vediamo.

La modalità login

- La modalità “login” della Bash la si ottiene quando il primo parametro passato all'eseguibile è “-” o “--login” (vedremo piu avanti)
- In questa modalità, la bash valuta ed esegue i seguenti files:
  - `/etc/profile`
    - Si tratta di un file di configurazione “system-wide”, valido per tutti gli utenti. Esegue una serie di configurazioni di base, come l'impostazione di “PS1” (il prompt di default) o rende non permanente la “command history” per l'utente root (ragioni di sicurezza).
    - Esegue in oltre una serie di altri script, contenuti nella directory `/etc/profile.d/` che ne consentono l'ampliamento (per esempio la bash_completion).

La modalità login

- `/etc/profile`
  - Tra le altre cose, è in questo file che viene settato il PATH, ovvero l'insieme delle cartelle in cui la bash cercherà un eseguibile a fronte di un comando.

    ```
    PATH=/bin:/usr/bin:/usr/local/bin
    ```

- `~/.bash_profile`
  - Questo è il “profile” personale dell'utente. Consente di estendere le impostazioni di `/etc/profile` (in sola lettura per gli utenti).
  - Il carattere “`~`” sta ad indicare la home directory dell'utente corrente (definita anch'essa nel file `/etc/passwd`).
  - Il file in questione quindi, è un file nascosto (preceduto dal carattere “.”), e situato nella home dell'utente che si è loggato.

La modalità login

  - `~/.bash_profile`
    - Nel caso in cui non esista questo file, bash proverà ad eseguire, nell'ordine, `~/.bash_login` e `~/.profile`
- Una volta che l'utente è loggato, avrà accesso allo storico dei comandi dati nelle precedenti sessioni (500 righe) che si trova nel file `~/.bash_history`
  - Su alcune distribuzioni, questa possibilità viene inibita per l'utente root per ragioni di sicurezza. Va detto però che la directory `/root/` (home directory dell'utente root) è inaccessibile a qualsiasi altro utente, la sua command history è al sicuro già di per se.
- Nel caso in cui sia presente il file `~/.bash_logout`, bash ne esegue il contenuto al termine della sessione (logout). E' molto utile, ad esempio, per pulire lo schermo (tramite `clear`) nel momento in cui si esegue il logout.

La modalità interattiva

- La modalità “interattiva” della Bash la si ottiene quando non viene passato alcun parametro, o con il parametro “-i”
- In questo caso viene valutato ed eseguito il solo `~/.bashrc`
- Spesso e volentieri tutti questi script vengono “interconnessi” tra di loro, magari attraverso dei link simbolici, in modo da “accentrare” tutta la configurazione in un unico file.

- Per compatibilità con il passato, se bash viene lanciata con il nome “sh”, si comporta come la vecchia shell “sh”, eseguendo i files `/etc/profile` e `~/.profile`

Switch User

- La differenziazione tra le due modalità che abbiamo descritto viene messa in risalto quando si vuole cambiare utente su una console, tramite l'uso del programma “su” (switch user)
  - `su - nomeutente`
    - Il parametro che verrà passato alla bash è “-”, quindi si tratta di una login shell. Vengono letti /etc/profile e ~/.bash_profile, portando di fatto ad un prompt identico a quello che si otterrebbe facendo un nuovo login con utente “nomeutente”
    - Ad esempio, viene modificato (da /etc/profile) il PATH, ma ci si sposta anche nella ~ di “nomeutente”.
    - Si tratta, di fatto, di un nuovo login.

Switch User

- La differenziazione tra le due modalità che abbiamo descritto viene messa in risalto quando si vuole cambiare utente su una console, tramite l'uso del programma “su” (switch user)
  - `su nomeutente`
    - Non vengono passati parametri alla bash, e quindi si tratta di una shell interattiva. Viene eseguito il solo `~/.bashrc`, e si mantiene l'ENVIRONMENT corrente, pur avendo i permessi di “nomeutente”
    - Si rimane nella directory in cui ci si trovava, si mantiene il PATH corrente, e cosi via.
    - Si perdono di fatto tutte le configurazioni fatte da `/etc/profile`, anche nel caso in cui `~/.bashrc` sia un link simbolico al file `~/.bash_profile`

Il prompt

- Una volta che tutti i file di startup sono stati eseguiti ed hanno terminato, all'utente viene mostrato il prompt.
- Questo viene “creato” valutando la variabile d'ambiente PS1

```
PS1=[\u@\h \W]\$
```

- Può sembrare arabo, ma non lo è, vediamo con calma.

Il prompt

- Una volta che tutti i file di startup sono stati eseguiti ed hanno terminato, all'utente viene mostrato il prompt.
- Questo viene “creato” valutando la variabile d'ambiente PS1

```
PS1=[\u@\h \W]\$
```

- Può sembrare arabo, ma non lo è, vediamo con calma.
  - `\u` Viene sostituito dal nome dell'utente.

Il prompt

- Una volta che tutti i file di startup sono stati eseguiti ed hanno terminato, all'utente viene mostrato il prompt.
- Questo viene “creato” valutando la variabile d'ambiente PS1

```
PS1=[\u@\h \W]\$
```

- Può sembrare arabo, ma non lo è, vediamo con calma.
  - `\u` Viene sostituito dal nome dell'utente.
  - `\h` Viene sostituito dal nome della macchina

Il prompt

- Una volta che tutti i file di startup sono stati eseguiti ed hanno terminato, all'utente viene mostrato il prompt.
- Questo viene “creato” valutando la variabile d'ambiente PS1

```
PS1=[\u@\h \W]\$
```

- Può sembrare arabo, ma non lo è, vediamo con calma.
  - `\u` Viene sostituito dal nome dell'utente.
  - `\h` Viene sostituito dal nome della macchina
  - `\W` Viene sostituito dalla directory corrente (working dir.)

Il prompt

- Una volta che tutti i file di startup sono stati eseguiti ed hanno terminato, all'utente viene mostrato il prompt.
- Questo viene “creato” valutando la variabile d'ambiente PS1

```
PS1=[\u@\h \W]\$
```

- Può sembrare arabo, ma non lo è, vediamo con calma.
  - `\u` Viene sostituito dal nome dell'utente.
  - `\h` Viene sostituito dal nome della macchina
  - `\W` Viene sostituito dalla directory corrente (working dir.)
  - `\$` Viene sostituito da un carattere che identifica la tipologia di utente: $ utente non privilegiato, # root.
- Il risultato è:

```
[alt-os@ganesh ~]$
```

Il prompt

- Esistono naturalmente altre sequenze di escape, tra le quali:
  - `\t` Orario corrente (hh:mm:ss)
  - `\d` Data
  - `\s` Nome della shell
- Il PS1 di default è "`\s-\v\$`" che espande al ben noto

  `bash-3.2$`

  che tanti utenti detestano.
- Il PS1 può essere cambiato anche dopo il login (fino al termine della sessione), semplicemente digitando

  `PS1=[...]`

  e battendo invio (di fatto, cambiando il valore della variabile d'ambiente).

Gli alias

- Uno dei contenuti classici degli script di startup della bash (soprattutto di `~/.bashrc` e `~/.bash_profile`) è l'impostazione degli “alias”.

```
alias ll='ls -la'
```

- Introduce un “comando” `ll` che di fatto esegue `ls -la` (mostrando anche i files nascosti che ls normalmente non mostra).
- Altri esempi di alias utili:

```
alias cd..='cd ..'          # As DOS does...
alias cp='cp -i'            # Prompt before override
alias d='ls'                # As DOS does...
alias l='ls'                # Broken 's' key?
alias rm='rm -i'            # Ask before remove...
```