

Introduzione a Linux e primi
passi per la realizzazione di una
minidistribuzione su pendrive:

MVux

Vittoria Cozza

April 10, 2008

Le directory di Linux

/: è la directory principale (o directory di root). Contiene il kernel predefinito.

bin: contiene tutti i programmi per la gestione del sistema.

boot: qui sono presenti i file di startup e i kernel alternativi (vmlinuz) e altri file utilizzati dal grub.

dev: in questa directory son contenuti i file di device, ossia i file necessari per l'utilizzo dei vari dispositivi (monitor, stampanti, dischi, etc). I dispositivi son visti come dei file speciali; un file speciale contenuto in dev è null, paragonabile ad un cestino di dimensione infinita: qualunque cosa venga scritta in questo file viene scartata.

etc: qui son presenti tutti i file di configurazione del sistema, del grub e il database degli utenti (dove son registrate le userid e la password di ciascun utente); ricorda il pannello di controllo di windows.

home: contiene le directory personali di tutti gli utenti.

initrd: è presente solo in alcune distribuzioni e contiene informazioni per la fase di boot.

lib: contiene tutte le librerie condivise del sistema.

lost+found: qui vengono raccolte le parti di file recuperate dopo una interruzione della corrente elettrica.

mnt: directory di mount.

proc: contiene un file system virtuale, creato dal kernel dinamicamente in memoria e non sul disco. Al suo interno son contenute informazioni relative al sistema.

Originariamente /proc è stata concepita come finestra in cui lo spazio utente potesse vedere alcune delle strutture interne del kernel, in particolare l'albero dei processi e tutte le loro caratteristiche, da qui il nome proc.

opt: contiene applicazioni particolari quali KDE, GNOME, altro.

root: è la directory home dell'amministratore di sistema.

sbin: contiene programmi di sistema eseguibili solo dall'amministratore del sistema come ad esempio fdisk o fsck.

tmp: contiene dei file temporanei.

usr: contiene tutte le applicazioni ed i file che servono per la gestione del sistema (come X-Windows, Latex, etc.). Qui vengono installati i vari pacchetti software.

var: è la directory dei file variabili. Contiene ad esempio i log files, i file di spool, l'immagine di un cd prima che sia masterizzato, etc.

All'interno della directory `usr` sono presenti solitamente le seguenti directory:

X11R6: contiene i file del sistema X-Windows.

doc: contiene i file di documentazione varia.

include: contiene i file header per il preprocessore C e C++.

local: contiene tutte le applicazioni e i file locali.

man: contiene il manuale online di Linux.

share: contiene i file indipendenti dall'architettura suddivisi per architetture.

src: contiene i sorgenti dei programmi del sistema, `/src/linux` contiene i sorgenti del kernel.

spool: esiste per motivi di compatibilità, in realtà contiene un link a `/var/spool`.

sbin: file binari di sistema secondari, ad esempio demoni per servizi di rete.

bin: comandi binari secondari (non necessari in single user mode).

lib librerie per i file binari contenuti in `/usr/bin/` e `/usr/sbin/`.

Principali comandi per i file

Ls cd pwd

chmod chgrp

Esempio

```
debianvittoria:~/Desktop/lez1# ls -l
```

```
totale 4
```

```
drwxr-xr-x 2 root root 4096 2008-04-03 10:30 es1
-rw-r--r-- 1 root root 0 2008-04-03 10:15 pippoh
lrwxrwxrwx 1 root root 5 2008-04-03 10:16 ps -> pippo
-rw-r--r-- 1 root root 0 2008-04-03 10:15 pluto.txt
-rw-r--r-- 1 root root 0 2008-04-03 10:15 topo.txt
```

I tipi di file sono:

file regolare

d directory

l link (collegamento)

b periferica a blocchi con buffer

c periferica a caratteri con buffer

u periferica a caratteri senza buffer

p pipe FIFO

s socket

N.B. i comandi chmod, chown e chgrp non hanno effetto su filesystem di tipo FAT

I dispositivi ed il comando mount
mount, umount, supermount daemon

```
mount -t vfat /dev/sda1
```

File System Tab (fstab file)

```
fstab
```

```
proc /proc proc defaults 0 0  
/dev/sda1 / ext2 defaults 1 1
```

Creare una partizione:
fdisk, cfdisk, mkfs.ext2

Esempio per formattare una pendrive usb per MVux

- Lanciare il comando `cfdisk /dev/sda` (`sdb` o `sdc` o ... nel caso siano presenti altre periferiche usb o sata)
- Creare una partizione delle dimensioni desiderate di tipo Linux (83).
- Confermare i cambiamenti effettuati tramite la scrittura sul disco.
- Uscire da `cfdisk`
- Lanciare il comando `mkfs.ext2 /dev/sda` (`mkfs.ext3`) per creare una filesystem sul hard disk rimovibile
- Creare la cartella come punto di montaggio `/mnt/hd`
- Montare l'hd appena formattato tramite `mount -t ext2 /dev/sda1 /mnt/hd`
- Entrare nella cartella `/mnt/hd`

In fase di avvio del sistema viene eseguito il comando `mount -a` presente nel file `/etc/rc.d/rc.sysinit` (oppure `rc.local`). In questo file sono contenuti tutti i comandi che vengono eseguiti in fase di avvio del sistema. Il comando `mount -a` viene a conoscenza dei dispositivi da collegare leggendoli dal file `fstab`.

ESEMPIO

```
#!/bin/sh
#
# local fs - check and mount local filesystems
#
PATH=/sbin:/bin ; export PATH

fsck -ATCp
if [ $? -gt 1 ]; then
echo \Errori presenti nel filesystem! Si richiede
intervento manuale."
/bin/sh
else
echo \Rimontaggio di / in modalit?a lettura-scrittura."
mount -n -o remount,rw /
echo -n >/etc/mtab
mount -f -o remount,rw /
echo \Montaggio del filesystem locale."
mount -a -t nonfs,nosmbfs
fi
#
# end of local fs
```

L'archiviazione dei file: i comandi tar (Tape ARchive) e gzip

Estrazione:

```
tar xf mioarchivio.tar
```

Compressione:

```
tar cfz mioarchivio.tar pippodir
```

Decompressione e estrazione:

```
gzip -d mioarchivio.tar.gz (oppure gunzip mioarchivio.tar.gz)  
e successivamente:tar -xf mioarchivio.tar
```

```
#!/bin/sh  
for i in *.tar.gz;  
{tar xvf $i}  
for j in *.tar.bz2;  
{ tar xjvf $j }
```

File di dispositivo

- fd0 Primo lettore di dischetti
- fd1 Secondo lettore di dischetti
- hda Disco fisso / CD-ROM IDE sulla prima porta IDE (Master)
- hdb Disco fisso / CD-ROM IDE sulla prima porta IDE (Slave)
- hdc Disco fisso / CD-ROM IDE sulla seconda porta IDE (Master)
- hdd Disco fisso / CD-ROM IDE sulla seconda porta IDE (Slave)
- hda1 Prima partizione sul primo disco fisso IDE
- hdd15 Quindicesima partizione sul quarto disco fisso IDE
- sda Disco fisso SCSI con l'ID SCSI pi basso (p.e. 0)

- sdb Disco fisso SCSI con l'ID SCSI successivo (p.e. 1)
- sdc Disco fisso SCSI con l'ID SCSI successivo (p.e. 2)
- sda1 Prima partizione sul primo disco fisso SCSI
- sdd10 Decima partizione sul quarto disco fisso SCSI
- sr0 CD-ROM SCSI con l'ID SCSI piu basso
- sr1 CD-ROM SCSI con l'ID SCSI successivo
- ttyS0 Porta seriale 0, COM1 sotto DOS
- ttyS1 Porta seriale 1, COM2 sotto DOS
- psaux Mouse PS/2
- cdrom Link simbolico al lettore CD-ROM
- mouse Link simbolico al file di device del mouse
- null Tutto ci che viene reindirizzato a questo device finisce nel nulla
- zero Da questo device si possono leggere infiniti zero

Link simbolici e hard link

```
debianvittoria:~/Desktop# cd lez1/
debianvittoria:~/Desktop/lez1# vi pippo.txt
debianvittoria:~/Desktop/lez1# vi pluto.txt
debianvittoria:~/Desktop/lez1# vi topolino.txt
debianvittoria:~/Desktop/lez1# ls
pippo.txt  pluto.txt  topolino.txt
debianvittoria:~/Desktop/lez1# ls -l
2012374 pippo.txt  2012375 pluto.txt
      2012376 topolino.txt
debianvittoria:~/Desktop/lez1# ln -s pippo pippos
debianvittoria:~/Desktop/lez1# ls
pippos  pippo.txt  pluto.txt  topolino.txt
debianvittoria:~/Desktop/lez1# ls -l
2012373 pippos  2012374 pippo.txt
      2012375 pluto.txt  2012376 topolino.txt
debianvittoria:~/Desktop/lez1# ln pippo.txt pippoh
debianvittoria:~/Desktop/lez1# ls -l
2012374 pippoh  2012374 pippo.txt
      2012376 topolino.txt  2012373 pippos
      2012375 pluto.txt
debianvittoria:~/Desktop/lez1#
```

Linux Kernel

- Scaricare i (file) sorgente del kernel dal sito <http://www.kernel.org/pub/linux/kernel/>
- Decomprimere i sorgenti in `/usr/src`

Del kernel son disponibili diverse versioni, esse sono numerate tramite un identificatore costituito da tre cifre, separate tra di loro da dei punti.

Il primo numero è il major version number (le versioni sviluppate finora appartengono alla serie 0, 1 o 2), e viene incrementato generalmente solo in caso di modifiche sostanziali al kernel.

Il secondo numero è il minor version number, che può essere pari o dispari: nel primo caso indica che si tratta di una versione giudicata stabile, mentre il secondo è la versione che viene periodicamente modificata dagli sviluppatori, ed è quindi più sperimentale e di conseguenza, non essendo testata completamente, più instabile.

Il terzo numero è utilizzato per indicare il livello di *patch* corrente: la *patch* è una modifica al kernel che in genere aggiunge nuove funzionalità o corregge i bachi riscontrati.

Per la realizzazione di MVUx faremo riferimento alla versione del kernel 2.6.23.

Sorgenti Linux

La suddivisione dei sorgenti di Linux nelle varie sottodirectory segue questo schema generale, indipendente dalle varie versioni:

kernel: parte relativa alla gestione generale del sistema.

mm: parte della gestione della memoria.

net: parte di gestione dei protocolli di rete.

fs: parte di gestione del file system.

drivers: parte dei drivers specifici dei periferici.

arch: parte contenente codice dipendente dall'architettura (es. i386).

/arch/boot: parte di basso livello per l'inizializzazione del sistema.

/arch/kernel: parte del kernel dipendente dall'architettura.

/arch/mm: parte del memory management dipendente dall'architettura.

/arch/lib: parte contenente funzioni di utilità .

/arch/math-emu: parte contenente il codice di emulazione del coprocessore.

include: file di include per i sorgenti, suddivisi in due parti.

/include/linux: file generali di Linux.

/include/asm: file dipendenti dall'architettura (es. asm-i386).

init: parte di alto livello per l'inizializzazione del sistema.

ipc: parte della gestione dell'Inter Process Communication.

lib: parte comprendente alcune funzioni di utilità .

modules: parte di gestione dei moduli caricabili a runtime nel kernel.

scripts file di shell per la configurazione, ad esempio tramite menu.

Inoltre è presente una directory **Documentation** che contiene diverse spiegazioni sulle operazioni basilari per far funzionare il kernel.

Per comprendere i sorgenti, è fondamentale una buona conoscenza del linguaggio C, in quanto questi sono spesso scritti in maniera non facilmente comprensibile a prima vista, causata sovente dal desiderio di ottenere una buona ottimizzazione da parte del compilatore.

Infine è utile possedere qualche rudimento di linguaggio assembler, in particolar modo di quello AT&T, che differisce un pò dal consueto linguaggio Intel: ciò è particolarmente utile per visionare le routine di basso livello, quali quelle di inizializzazione delle tabelle di segmentazione e paginazione, di impostazione dei vari bit nei registri del processore, nonché il codice del context switching.

Compilazione del kernel

- Aprire la cartella `/usr/src/Linux-source-f` versione g
- Lanciare la configurazione del kernel tramite uno dei comandi: `make menuconfig`, `make xconfig`, `make qconfig`
- Includere nel kernel, non come moduli, le componenti relative al riconoscimento delle periferiche usb o sata
- Includere nel kernel le componenti relative ai filesystem ext e alle periferiche di input come mouse e tastiera
- Salvare la configurazione appena creata (di default nel file `.config`)
- Lanciare il comando `make`
- Lanciare il comando `make bzImage`
- Lanciare il comando `make modules`

Il comando `make bzImage` ha l'obiettivo di creare un eseguibile del kernel `bzImage`, collocato nella cartella `/usr/src/linux-source/arch/i386/boot/` compresso tramite il formato `bz`.

Se necessario dopo la compilazione è utile lanciare il comando `make modules install` il quale avrà come output la creazione della cartella `/lib/modules/fversione` del kernel al cui interno sarà presente l'albero dei moduli appena compilati.

Per capire quali moduli includere nel kernel nella fase di configurazione, è fondamentale l'uso del comando `lspci` che elenca i dispositivi collegati tramite la porta `pci`.

Compilazione di Bash e installazione nella pendrive

Scaricare i sorgenti dal sito <http://ftp.gnu.org/gnu/bash>

- Entrare nella cartella `/usr/src/bash-3.2`
- Lanciare il comando `./configure` per verificare la configurazione del sistema
- Lanciare il comando `make` per compilare i sorgenti
- Verificare la presenza dell'eseguibile `bash` nella cartella
- Entrare nella cartella `/usr/src/linux-source - versione/arch/i386/boot`
- Copiare il file `bzImage` qui presente nella cartella `boot` dell'hard disk rimovibile cambiando il nome in `vmlinuz` tramite il comando

```
cp bzImage /mnt/hd/boot/vmlinuz
```
- Copiare l'eseguibile di `bash` compilato nella cartella `bin` dell'hard disk rimovibile
- Entrare nella shell
- Posizionarsi nella cartella dove è presente l'eseguibile di `bash`

- Lanciare il comando `ldd ./bash`; A questo punto verrà visualizzato un elenco di librerie necessarie per l'esecuzione di bash ciascuna col proprio **PATH**.
- Copiare tutte le librerie rispettando il **PATH** sull'hard disk rimovibile
- Creare un link simbolico a bash tramite il comando `ln -s bash sh`

N.B. Nei primi sistemi, un interprete dei comandi era implementato built-in del sistema come modulo del kernel. Multics introdusse l'idea (poi diffusa da Unix) di interpreti dei comandi che son separati dal kernel e son invocati come programmi di utilità .

Boot loader

La maggior parte dei calcolatori possono solo eseguire il codice che risiede in memoria (ROM e RAM). Sistemi operativi più moderni si trovano però su hard disk, in LiveCDs, USB flash drives, e altri dispositivi di memorizzazione non volatile. Non appena un computer viene acceso, esso non ha in memoria un sistema operativo, l'hardware da solo non è in grado di svolgere il compito complesso di caricare un programma da un disco... allora sembra che per caricare in memoria un sistema operativo ci sia bisogno di aver un sistema operativo già caricato!

La soluzione è quella di usare un piccolo programma speciale chiamato *bootstrap loader*, *bootstrap* o *boot loader*.

Il compito di questo programma è esclusivamente quello di caricare il software necessario per permettere l'avvio del sistema operativo.

Il bootloader dunque fornisce la possibilità di definire gli eseguibili da avviare e di poter specificare delle opzioni su questi eseguibili. Inoltre è possibile anche specificare su quale harddisk e su quale partizione risiedono questi eseguibili.

Alcuni bootloader noti per Linux sono:

- **LILO (Linux LOader)**: si installa in una partizione di un hard disk, nel Master Boot Record, in un floppy disk.
- **GNU GRUB (GRand Unified Boot loader)**: è un potente boot loader che può avviare una ampia

varietà di sistemi operativi sia opensource sia proprietari. Molto apprezzato per la sua flessibilità , infatti GRUB sa lavorare con i filesystems e i kernel in formato eseguibile, perciò puoi caricare un sistema operativo, senza scrivere la posizione fisica del kernel sul disco; infatti per caricare il kernel basta specificare il nome del file, il drive e la partizione dove il kernel risiede.

Oltre al GRUB boot loader c'è un grub shell che si può avviare nel proprio sistema operativo e serve per emulare il boot loader e può essere usata per installarlo.

- **LOADLIN:** permette di avviare Linux da linea di comando DOS (hard-reset).
- **etc.**

Compilazione del Grub e installazione su pendrive

- Entrare nella cartella `/usr/src/grub-0.97`
- Lanciare il comando `make` per la compilazione del codice sorgente
- Verificare la presenza dei file `stage1` e `stage2` nella cartella
- Entrare nella sottocartella `/usr/src/grub-0.97/grub`
- Verificare la presenza dell'eseguibile chiamato `grub`
- Copiare all'interno della cartella `boot/grub` dell'hard disk rimovibile i file `stage1` e `stage2` presenti nelle cartelle `/usr/src/grub-0.97stage1` - `stage2`
- Copiare i file `*_stage1_5` dalla cartella `./grub-97/stage2` alla cartella `/mnt/hd/boot/grub`
- Avviare `grub` tramite il comando `grub` da digitare nella shell di comando.
- Lanciare il comando `root` (e tramite il tasto `TAB` verificare le disponibilità e scegliere l'hard disk rimovibile.

- Lanciare il comando setup (hd1)
- quit per uscire dal grub.

I file che copiamo nella cartella /boot/grub son degli eseguibili particolari che si riferiscono agli stadi in cui è suddivisa la fase di avvio. Il file stage1 è lungo esattamente 512 byte, ovvero un settore: questo è il primo pezzo di codice che viene utilizzato durante l'avvio.

Il file stage2 rappresenta il codice necessario al completamento dell'avvio, gli altri file si riferiscono ad una fase intermedia. Alcuni hanno come prefisso il nome di un file system, bisogna selezionare quelli in uso sul computer dal quale si farà partir il sistema.

Il comando root serve per definire quale partizione contiene i file per l'avvio, mentre il comando setup contiene solo l'identificativo del disco su cui il MBR andrà ad installare il boot loader.

Primo avvio di MVux

- Spegnere il pc
- Inserire l'harddisk rimovibile
- Avviare il pc ed entrare nel bios (solitamente con DEL o F12 o F2)
- Entrare nel menu Boot
- verificare la presenza del proprio hard disk rimovibile fra i dischi rigidi e impostare la priorità di boot come primaria per l'hd stesso
- Nel caso in cui l'hd non sia presente tra i dischi rigidi impostare come primo dispositivo di boot le periferiche rimovibili seguite dagli hard disk.
- Salvare le modifiche ed uscire dal bios (solitamente con F10)
- Attendere l'avvio del pc dall'hard disk rimovibile
- Verrà mostrata la riga di comando del grub, dunque lanciare il comando *root(hd0,0)*

- Lanciare il comando kernel con i parametri

```
kernel /boot/vmlinuz root=/dev/sda1  
rootdelay=10 init=/bib/bash
```

- Lanciare il comando boot

Il parametro di configurazione `rootdelay` si riferisce al numero di secondi che deve attendere il kernel prima di cercare la root.

Dopo aver caricato il kernel in memoria e dopo aver riconosciuto il file system della partizione di root, è necessario specificare il processo iniziale tramite l'opzione `init`.