



Il boot del sistema

Daniele lamartino (aka オタコン22)
<secretario@poul.org>



Quando il sistema operativo non è ancora pronto, il sistema deve riuscire a farcela da solo, "tirandosi su le cinghie dei propri stivali"

Il BIOS

- È quel chip integrato sulla scheda madre del computer, con un piccolo programma per le operazioni elementari.
- I suoi scopi principali sono:
 - Controlli hardware e rilevazione dischi
 - Gestire le priorità di boot
- È configurabile tramite una schermata che appare premendo una qualche combinazione speciale di tasti subito dopo aver acceso.
- “Don't panic”, solitamente insieme alla scheda madre c'è un manuale per decifrare cosa digitare/fare nel BIOS.

II BIOS: caso tipico

 **American
Megatrends** AMIBIOS (C)2001 American Megatrends Inc.,
K7S5A Release 09/20/2001 S

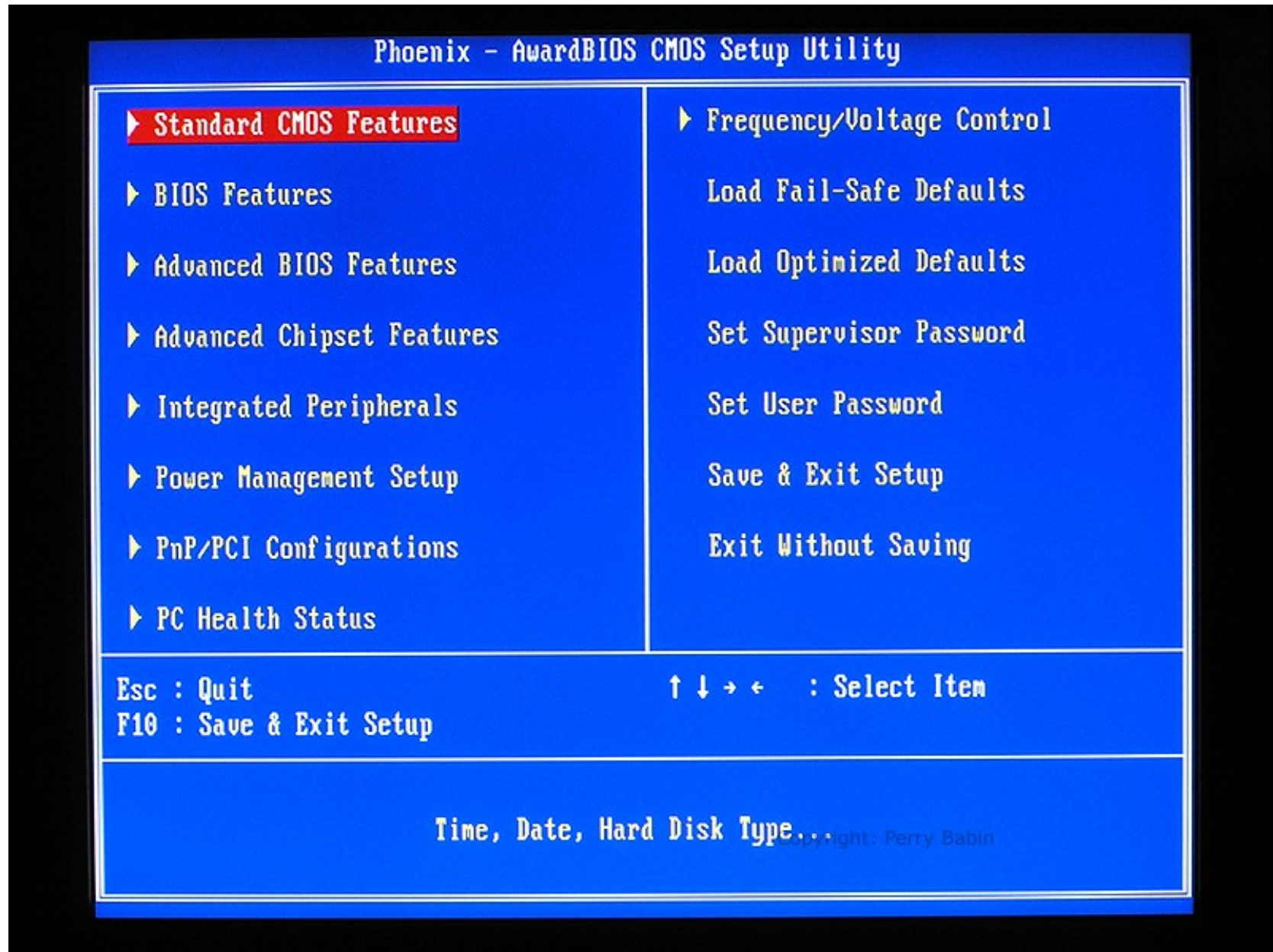


AMD Athlon(tm) XP 1800+
Checking NVRAM..
262144KB OK_

Go->Setup F8:Boot Menu F12:Network boot ESC:Skip memory test

(C) American Megatrends Inc.,
62-0920-001437-00101111-040201-SiS735-K7S5A-

II BIOS: caso tipico



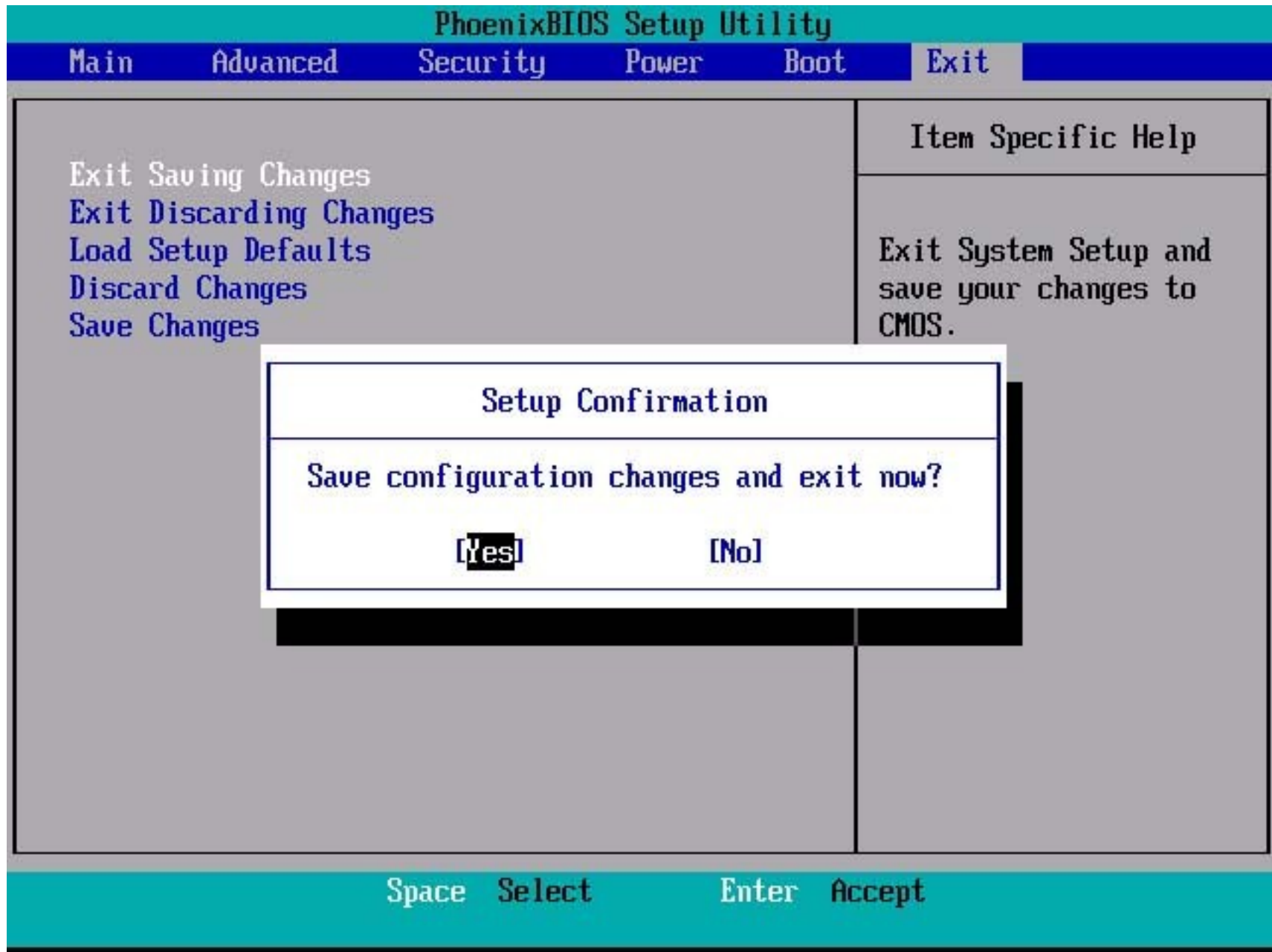
II BIOS: caso tipico

Phoenix - AwardBIOS CMOS Setup Utility
Boot Setting Configuration

▶ HDD Boot Priority	Press Enter	▲	Item Help
▶ Removable Boot Priority	Press Enter		
▶ CD-ROM Boot Priority	Press Enter		Menu Level ▶
1st Boot Device	Hard Disk		
2nd Boot Device	CDROM		Select Hard Disk Boot Device Priority
3rd Boot Device	Removable		
Boot Other Devices	Enabled		
Boot Up NumLock Status	On		
Gate A20 Option	Fast		
Security Option	Setup		
APIC Mode	Enabled		
MPS Version Control For OS	1.4		
OS Select For DRAM > 64MB	Non-OS2		
Report No FDD For WIN 95	No		
HDD Detection Delay (s)	0		
Display Full Screen Logo	Enabled		
Display Quantum Logo	Enabled		
Display Summary Screen	Disabled		
Debug Code Control	LPC		
System BIOS Cacheable	Enabled	▼	

↑↓←→:Move Enter:Select +/-/PU/PD:Value F10:Save ESC:Exit F1:General Help
F5:Previous Values F7:Optimized Defaults

II BIOS: caso tipico



II BIOS: solo boot menu



Boot-loader

- Il BIOS legge i primi 512 bytes del disco di avvio (che sono anche chiamati MBR) e li utilizza come “boot-loader”, passandogli il controllo.
- Il boot-loader ci servirà per scegliere quale sistema operativo avviare e quindi caricare il **kernel** per proseguire nel boot del sistema operativo vero e proprio.
- Vecchi bootloader (ex. “Lilo”) riuscivano ad essere completamente contenuti nei 512 bytes dell'MBR, lì viene memorizzato anche il riferimento al kernel.
- Altri bootloader moderni (ex. GRUB) utilizzano l'MBR per caricare un'altra parte di se stessi (non riuscendo a restare in 512 bytes)

GRUB

- **Grub stage 1** risiede nell'MBR ed ha lo scopo di caricare da disco il secondo “pezzo” di se stesso: **Grub stage 2**
- Grub stage 1, al fine di caricare da disco, contiene al suo interno tutto il necessario per caricare i dati da una partizione (fat32, ext2, ext3, ReiserFS...)
- Recentemente è nata la **versione 2** di GRUB che permette tra le altre cose anche il supporto a più filesystems (ext4 e altri), LVM (vedremo più avanti nel corso) e cambia la configurazione.
- Grub “stage 1.5” che si legge a volte, è una parte di GRUB posizionata nei primi settori seguenti l'MBR

BIOS

GRUB stage 1

GRUB stage 2

GRUB: Stage 2

- Grub, ormai pronto, richiede all'utente quale sistema operativo avviare e c'è anche un countdown
- In caso di necessità c'è una sorta di terminale integrato per modificare opzioni di avvio “al volo”, a volte molto utile.

BIOS

GRUB stage 1

GRUB stage 2

Ubuntu 8.04.1, kernel 2.6.24-19-generic
Ubuntu 8.04.1, kernel 2.6.24-19-generic (recovery mode)
Ubuntu 8.04.1, kernel 2.6.24-18-386
Ubuntu 8.04.1, kernel 2.6.24-18-386 (recovery mode)
Ubuntu 8.04.1, kernel 2.6.24-16-386
Ubuntu 8.04.1, kernel 2.6.24-16-386 (recovery mode)
Ubuntu 8.04.1, kernel 2.6.24-16-generic
Ubuntu 8.04.1, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04.1, memtest86+
Other operating systems:
Microsoft Windows XP Professional

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

Configurazione di grub

- Da qualche parte nel file di configurazione di grub, o nella command line editabile all'avvio apparirà qualcosa del genere:

```
linux /vmlinuz-2.6.32-5-686 root=/dev/sda1 ro quiet  
initrd /initrd.img-2.6.32-5-686
```

GNU GRUB version 0.95 (638K lower / 128960K upper memory)

```
root (hd0,0)
kernel /boot/vmlinuz-2.6.15-1-686 root=/dev/sda1 ro
initrd /boot/initrd.img-2.6.15-1-686
savedefault
boot
```

Use the ↑ and ↓ keys to select which entry is highlighted.
Press 'b' to boot, 'e' to edit the selected command in the
boot sequence, 'c' for a command-line, 'o' to open a new line
after ('O' for before) the selected line, 'd' to remove the
selected line, or escape to go back to the main menu.



Kernel Linux

- Una volta scelto il sistema operativo, GRUB si occupa di caricare da disco il kernel e mandarlo in esecuzione caricandolo in RAM.
- GRUB, nel lanciare il kernel gli passa alcune opzioni, come ad esempio quale disco dovrà usare il kernel come partizione “root” (root=...)
- Da qui in poi non importa chi ha avviato, del resto se ne occupa il kernel.
- Nota: il kernel viene caricato da GRUB a partire dalla cartella /boot (in partizione separata)
- Ogni volta che vorremo modificare il kernel dovremo quindi “avvertire” il bootloader, o comunque fargli trovare quello nuovo in una posizione nota.

BIOS

GRUB stage 1

GRUB stage 2

Kernel (Linux)

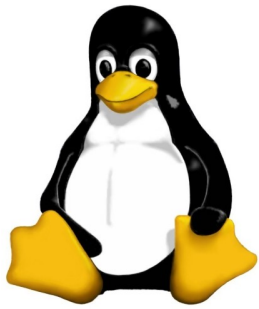


Kernel Linux

- Il kernel avviato come prima cosa si occupa di “montare” la partizione root “/”
- Se non dovesse esistere nessuna partizione root il kernel non potrà proseguire e sarà costretto a fermarsi in un “*kernel panic*”
- Se non ci sono problemi, la partizione root verrà caricata nella modalità specificata da GRUB
 - Sola lettura (configurazione classica)
 - Lettura e scrittura
- Se risulta esserci qualche errore sul disco, o nel caso il kernel lo ritenga necessario, viene effettuato un controllo del disco (fsck).
- Se non c'è nessun problema il disco viene rimontato in lettura e scrittura

C'è qualcosa che non va?

- C'è qualche problema con il filesystem
- Nel caso semplice il “file” del kernel contiene compilato al suo interno tutto il necessario per riconoscere e montare le partizioni root, ma non è sempre così.
- Riempire il kernel “di roba” è svantaggioso.
- Come fa il kernel a montare il filesystem se necessita di driver per leggerlo, e i driver sono sul disco stesso?!
- La soluzione è utilizzare **initrd** (*Initial ramdisk*) che è un file (solitamente compresso) contenente piccolo filesystem leggibile dal kernel che viene caricato in RAM. Esso contiene tutti i driver e moduli necessari per caricare la vera “/”
- Torna molto utile per:
 - Caricare moduli (drivers) essenziali per il controller del disco dove c'è la root (ad esempio: partizione di root su chiavetta USB)
 - Caricare moduli (drivers) necessari per leggere la partizione di root
 - Decifrare dischi cifrati
- Non è obbligatorio usarlo, alcuni lo sconsigliano preferendo compilare tutto il necessario nel kernel



Kernel Linux

- Una volta caricata la partizione di root, il kernel prosegue a identificare tutto **l'hardware** presente ed a caricare i **driver** (moduli del kernel) necessari per il suo utilizzo.
- Il kernel ha formalmente completato il boot e lancia il primo processo del sistema operativo: **init**, caratterizzato dal PID (Process Identifier) di valore 1.
- Insieme ad init vengono lanciati alcuni processi “spontanei” (threads del kernel)
- Il kernel ha completato il suo lavoro di avvio e il sistema operativo passa a “livello utente”

Init

“il grande padre” (cit.)

- Una volta lanciato **init**, il sistema operativo è formalmente pronto.
- Init ha il compito di generare tutti gli altri processi necessari al sistema. Esempio classico di processo generato da init è **getty** (attiva il terminale e inizia la procedura di accesso).
- Init, per sapere cosa fare, utilizza come proprio file di configurazione **/etc/inittab**
- **/etc/inittab** divide i vari programmi da lanciare all'avvio in livelli di esecuzione chiamati **“runlevel”**.

getty

```
Ubuntu 9.10 curso-desktop tty1
```

```
curso-desktop login: _
```

Runlevel e init

- L'avvio dei vari processi del sistema può essere suddiviso in 6 livelli di funzionamento
 - Runlevel 0 (speciale): sistema completamente fermo
 - Runlevel 1 o S: modalità monoutente (single user mode)
 - Runlevel da 2 a 5: modalità multiutente
 - Runlevel 6: riavvio
- /etc/inittab :
 - Indica ad init di lanciare e controllare che siano sempre attivi alcuni processi (ad esempio getty).
 - In generale definisce cosa fare per ciascun runlevel.
- All'avvio il sistema si porta dal runlevel 0 fino al runlevel di default (anche questo dichiarato in /etc/inittab come vedremo in seguito) passando in tutti quelli intermedi. Viceversa allo shutdown.

Runlevels in Debian



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

- Qualche volta il runlevel 5 viene utilizzato per i processi di login per le interfacce grafiche (ad esempio per xdm).
- Il runlevel 5 era stato introdotto per richiedere la password di root (...)

Script di avvio

- Ad ogni runlevel, init chiama uno script chiamato “rc”, che si trova in:

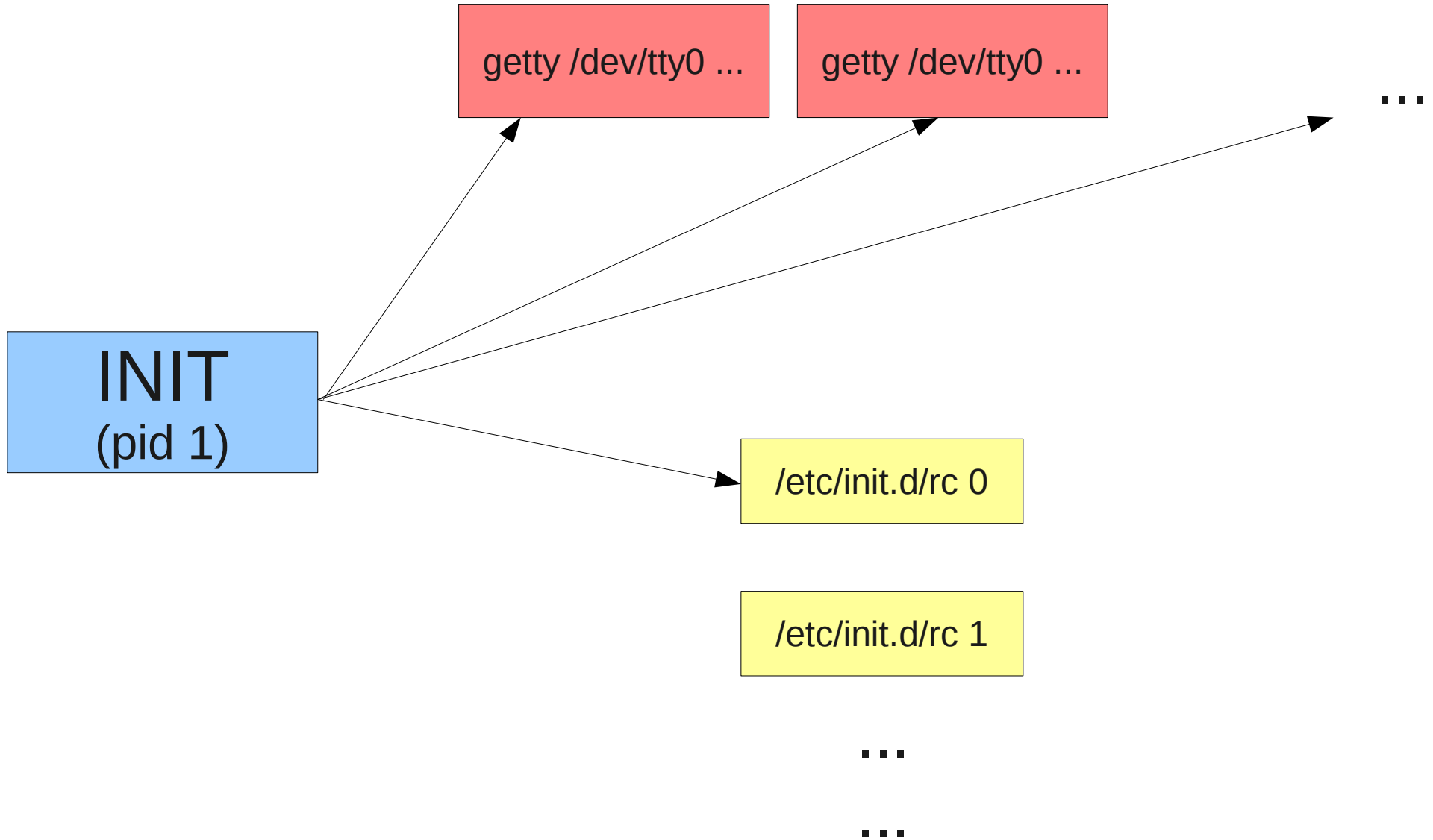
`/etc/init.d/rc`

- rc si occupa a sua volta di lanciare i programmi che servono per ogni runlevel
- Ad esempio in `/etc/inittab` può esserci qualcosa del genere:

```
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
...
```

Parametro del runlevel

Init



Avvio “alla System V”

- È stata la “filosofia” più diffusa per organizzare gli script di avvio, che prende il nome dal ramo di Unix che la implementava.
- Esiste una cartella per ogni runlevel:
 - /etc/rc0.d/
 - /etc/rc1.d/
 - ...
 - /etc/rc**N**.d/
- In queste cartelle ci sono una serie di **link** (simbolici) che puntano a files in /etc/init.d/
 - (esempio con ln)
- I files nelle cartelle di ogni runlevel hanno un nome strutturato in questi due modi:
 - /etc/rc0.d/**SX**nome
 - /etc/rc0.d/**KX**nome

Avvio “alla System V”

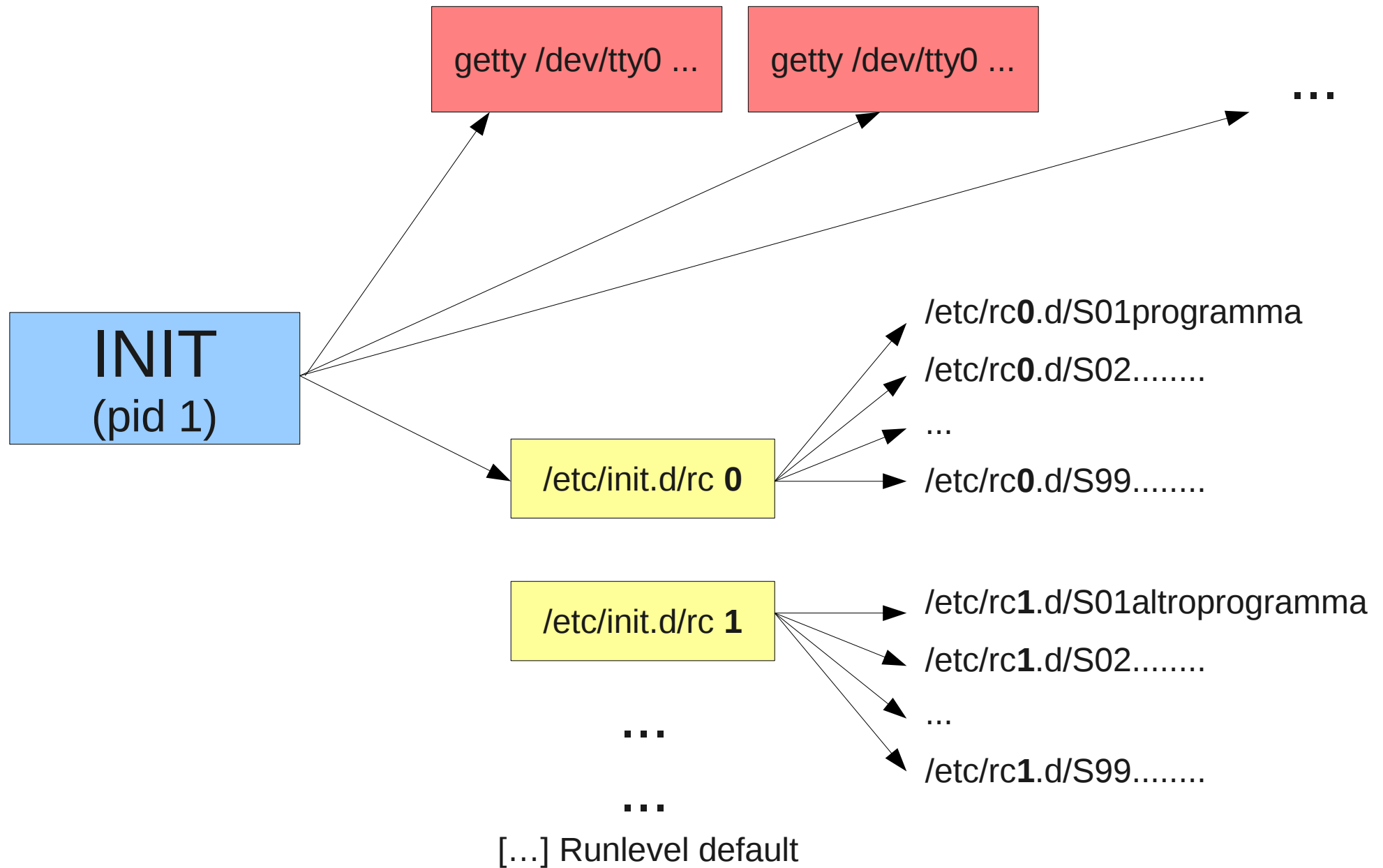
- Vediamo un esempio:

`/etc/rc0.d/S35networking` → `/etc/init.d/networking`

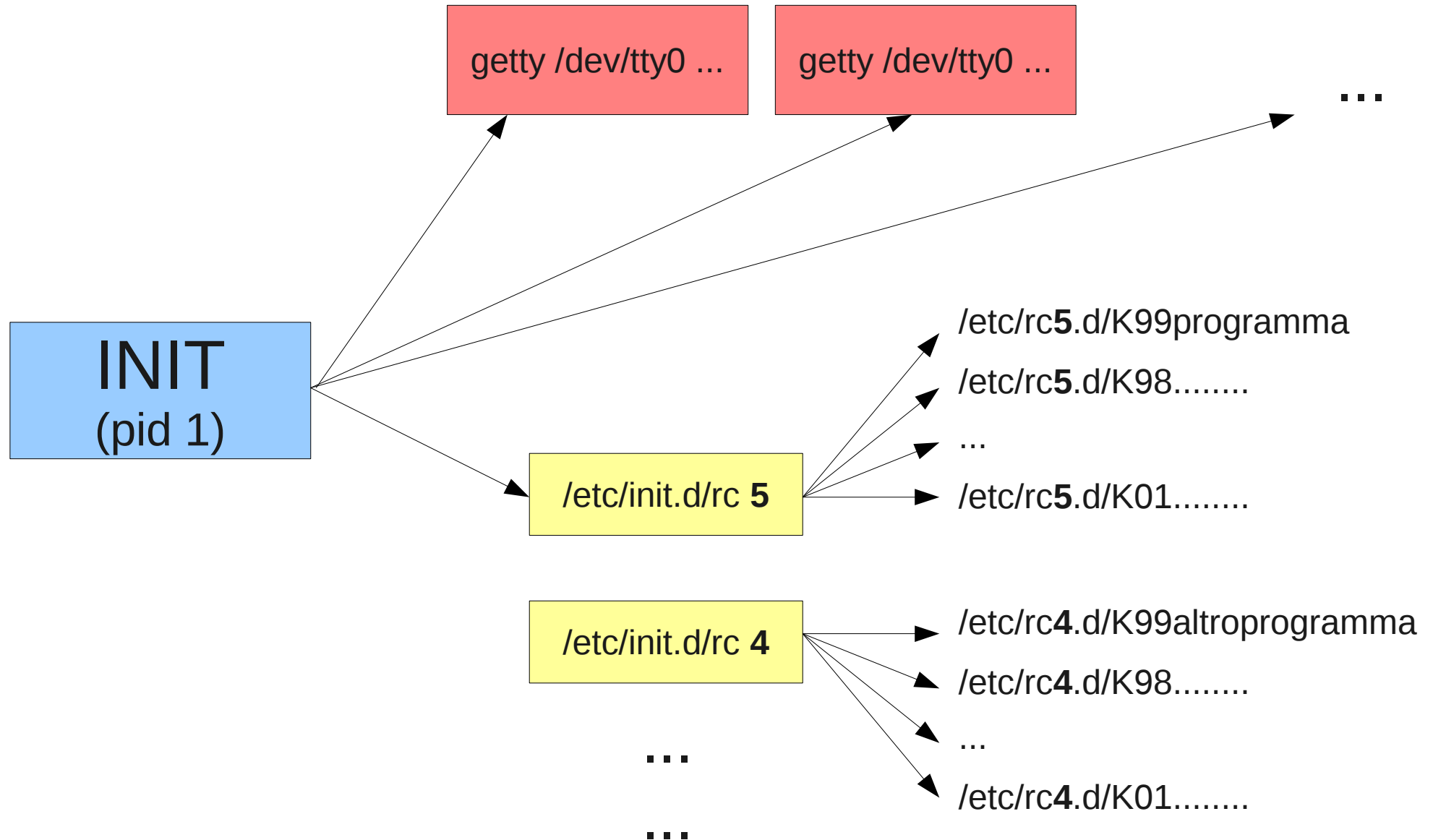
↑
“Start”, 35esimo

- (Esempio da terminale)

Init e rc (avvio del sistema)



Init e rc (arresto del sistema)



Link simbolici e demoni

- I file in `/etc/rcN.d/...` sono link simbolici a file contenuti in `/etc/init.d/`, cioè il loro percorso viene “riscritto”
- Ma i veri files in `/etc/init.d/` allora cosa contengono?
- Sono degli script che vengono interpretati da **bash**. Si occupano di un particolare aspetto del sistema o di avviare un **daemon**.
- Tra le attività solitamente eseguite da questi script c'è:
 - Impostare l'hostname
 - Impostare il fuso orario
 - Controllare i dischi con `fsck`
 - Montare i dischi del sistema
 - Rimuovere i vecchi file da `/tmp`
 - Avviare i **daemon** e i servizi di rete (ex. `Ntpd`, `apache`, `vsftpd` ...)
- *Un demone (daemon in inglese) è un programma eseguito in background, senza che sia sotto il controllo diretto dell'utente. Di solito i demoni hanno nomi che finiscono per "d" (cit.)*

Link simbolici (esempio)

```
#!/bin/sh
```

```
.....
```

```
case "$1" in  
start)
```

```
.....  
.....
```

```
stop)
```

```
.....  
.....
```

```
force-reload|restart)
```

```
.....  
.....
```

```
exit 0
```

Lancio degli script manualmente

- Per meglio comprendere il meccanismo di lancio ipotizziamo di voler manualmente avviare lo script che si occupa di avviare il server **web apache**

```
root@PhoenixDesktop:~# /etc/init.d/apache start
```

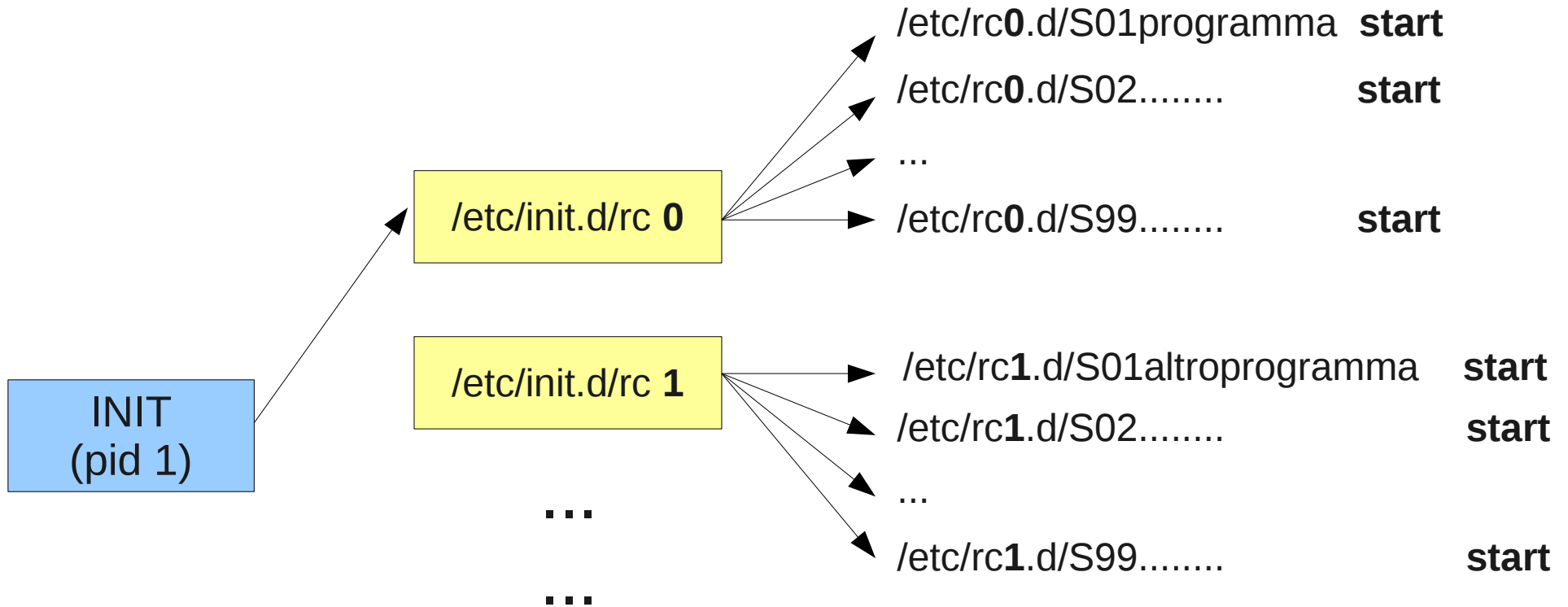
- Per arrestarlo:

```
root@PhoenixDesktop:~# /etc/init.d/apache stop
```

- Riavvio del demone/servizio:

```
root@PhoenixDesktop:~# /etc/init.d/apache restart
```

Lancio degli script di avvio



update-rc.d

- È un comando leggermente “clandestino” per gestire gli script di avvio.
- Si occupa di creare/eliminare i link simbolici nelle cartelle
- Vogliamo lanciare che sshd venga eseguito nei runlevel 2,3,4,5 e arrestato nei runlevel 0,1 e 6:

update-rc.d sshd start 2345 stop 016

- Per eliminare da tutti i runlevels:

update-rc.d -f avahi-daemon remove

- Potrebbe non essere sufficiente... (vedi aggiornamenti)

/etc/rc.local

- Solitamente è il file che viene eseguito per ultimo ed è vuoto.
- Se vogliamo lanciare qualche comando dopo che il sistema è completamente “avviato” possiamo usare questo file.

/etc/inittab

- La sintassi generale del file è
`<id>:<runlevels>:<action>:<process>`
- I campi sono:
 - **Id**: una stringa univoca che identifica l'operazione
 - **Runlevels**: l'elenco dei runlevels in cui va effettuata quella operazione
 - **Action**: l'azione da fare con quel processo (respawn, wait, ctrlaltdel)
 - Una action “speciale” è initdefault:
`id:5:initdefault:`

/etc/inittab (esempio)

The default runlevel.

id:2:initdefault:

Boot-time system configuration/initialization script.

This is run first except when booting in emergency (-b) mode.

si::sysinit:/etc/init.d/rcS

What to do in single-user mode.

~~:S:wait:/sbin/sulogin

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

Normally not reached, but fallthrough in case of emergency.

z6:6:respawn:/sbin/sulogin

What to do when CTRL-ALT-DEL is pressed.

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

1:2345:respawn:/sbin/getty 38400 tty1

2:23:respawn:/sbin/getty 38400 tty2

3:23:respawn:/sbin/getty 38400 tty3

4:23:respawn:/sbin/getty 38400 tty4

5:23:respawn:/sbin/getty 38400 tty5

6:23:respawn:/sbin/getty 38400 tty6

init

- Quando scriviamo da terminale (come root) il comando **init 1**
Stiamo dicendo ad init di “portarci” nel runlevel 1, cioè in modalità monoutente (single user mode).
- **init 0** arresta il sistema
- **init 6** riavvia il sistema

Avvio in Ubuntu - Upstart

- System V non è l'unica filosofia per organizzare gli script
- “Upstart” è un sostituto di init per l'avvio dei processi, inizialmente introdotto da Ubuntu (6.10)
- È in un certo senso retrocompatibile con System V
- Init c'è ancora ma si occupa solo delle sue funzioni di padre. /etc/inittab è scomparso.
- I vari script di avvio sono ancora contenuti in /etc/init.d/
- La cartella /etc/init/ contiene dei file di configurazione per ogni servizio, che specificano quando deve essere avviato (si evita di creare file simbolici)

Avvio “alla BSD” (Arch Linux)

- `/etc/init.d/rc` viene suddiviso spesso in diversi files come ad esempio:
 - `/etc/rc.sysinit` , `/etc/rc.single` , `/etc/rc.multi`
- Tutti gli script di avvio (e non i link) sono in `/etc/rc.d/`
- La scelta di quali script lanciare viene fatta in base a informazioni in `/etc/rc.conf`

Avvio in Red Hat e SUSE

- In Red Hat e Fedora, lo script **rc** si trova in `/etc/init.d/rc` e viene sempre lanciato passandogli come parametro il runlevel, come prima.
- L'avvio degli script/demoni/servizi può essere gestito con il comando **chkconfig** ed in generale la configurazione del processo di avvio è nella cartella `/etc/sysconfig`.
- OpenSUSE e altre distribuzioni (Fedora) stanno migrando al nuovo **systemd**.

Comandi per l'avvio e l'arresto

- Ci sono tanti modi per spegnere il sistema operativo, scegliamo quello più adatto.. Evitiamo di scomodare fsck (sempre se ce la fa..)
 - Stacciamo la spina (-.-') / premere “il pulsante”
 - Comando shutdown
 - Comandi halt e reboot, poweroff
 - Utilizzare init
 - SysRq per emergenze..

Shutdown

- Shutdown è il comando per spegnere una macchina nel modo più “pulito”.

shutdown -h now (per arrestare subito, h=“halt”)

shutdown -r now (per riavviare subito,
r=“reboot”)

- È possibile specificare l'orario di arresto/riavvio

shutdown -h 09:42

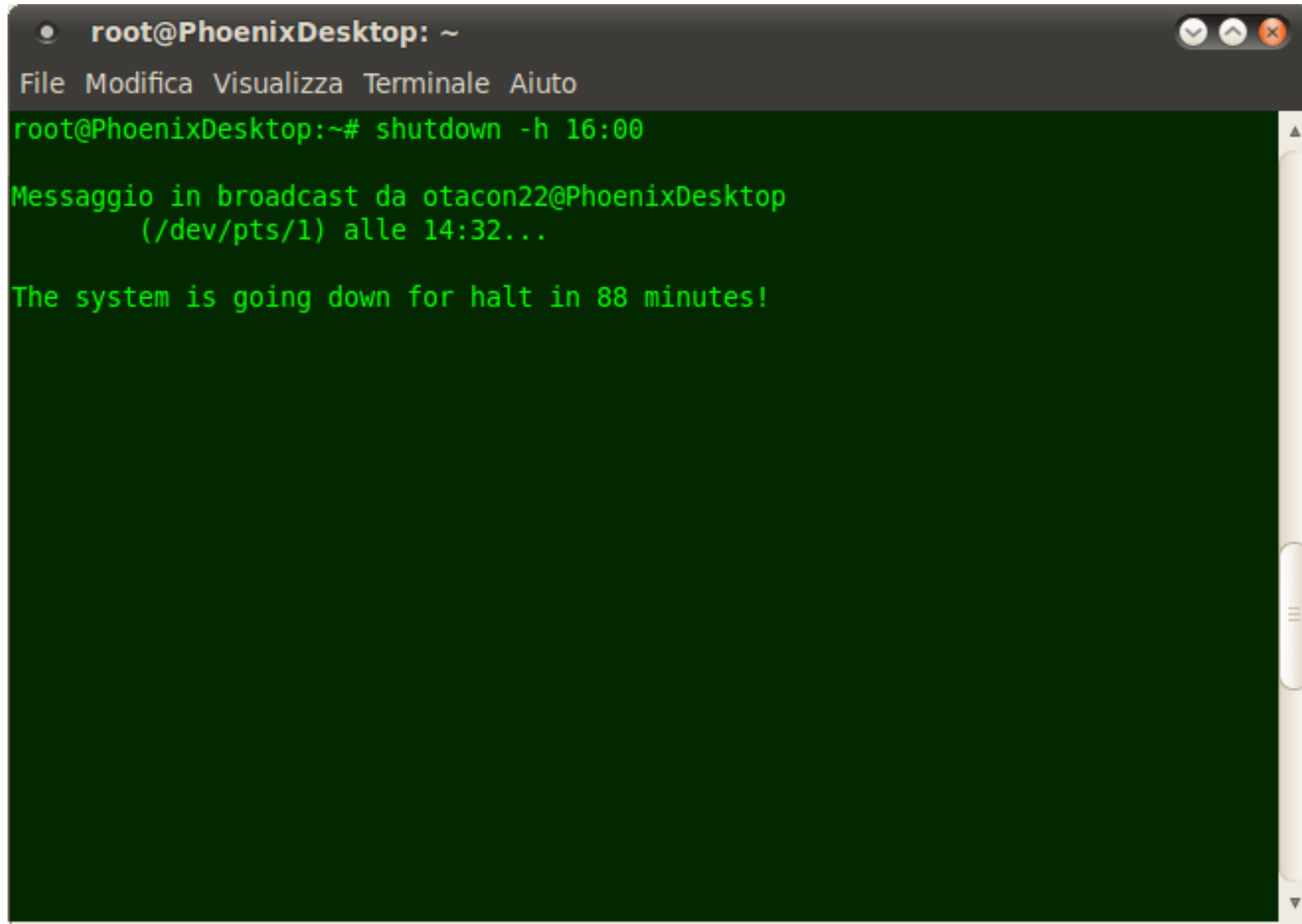
- Subito dopo l'esecuzione del comando viene inviato un avviso su tutti i terminali attivi che il sistema sta per arrestarsi (utile su macchine con utenti remoti).

- È possibile specificare un messaggio di avviso (wall)

shutdown -h 09:42 “Spegimento per manutenzione.”

- Possiamo annullare con: **shutdown -c**

Shutdown



```
root@PhoenixDesktop: ~  
File Modifica Visualizza Terminale Aiuto  
root@PhoenixDesktop:~# shutdown -h 16:00  
Messaggio in broadcast da otacon22@PhoenixDesktop  
  (/dev/pts/1) alle 14:32...  
The system is going down for halt in 88 minutes!
```

Halt e reboot

- I comandi **halt** e **reboot** sono quelli che richiama a sua volta il comando shutdown.
- Entrambi, prima di procedere all'arresto, fanno una chiamata di sync, per completare le operazioni I/O sui device.
- Esiste anche **poweroff** oltre ad halt. La differenza è che poweroff invia anche il segnale di interrompere l'alimentazione al termine (fa la differenza solo su alcuni sistemi server). Per usare poweroff con shutdown è possibile lanciare:

shutdown -P now

- In realtà -h può fare sia halt che poweroff, dipende dal sistema..
- Abbiamo già visto che init permette di portare il sistema nel runlevel corretto. Però non vengono mandati messaggi di preavviso e non possiamo impostare un orario..



SysRq

“Raising Elephants Is So Utterly Boring” (cit.)

- Se abbiamo accesso fisico ad un computer e c'è qualche problema molto grave, c'è un' alternativa a staccare la spina..
- Se almeno il kernel è ancora funzionante possiamo premere una sequenza di tasti per avvertire il sistema di terminare tutti i processi, **smontare i dischi** e quindi interrompere l'alimentazione o riavviare.

Alt + SysRq + R E I S U B

- In linguaggio umano:
 - Togli il controllo della tastiera al sistema grafico
 - Invia il segnale di arresto (SIGTERM) a tutti
 - Uccidi tutti i processi (SIGKILL)
 - Termina le operazioni di I/O (sync)
 - Smonta tutti i dischi (e rimonta in lettura)
 - Riavvia brutalmente.
- Non sempre sono abilitate di default

Bibliografia consigliata

- In generale per il corso:



Google libri

Grazie per l'attenzione!

ありがとうございます！

Domande?

Per altre informazioni/domande/whatever™

segretario@poul.org