Corso Linux Avanzato 2013 Terza Lezione 1/3

Init System (sistema di avvio) cosa fa?

-Avvia/riavvia il computer
-Termina l'avvio dopo che il kernel è stato caricato
-Avvia altri programmi (demoni). (EG. Server grafico, Server SQL o HTTP).

Recall: Demone "i demoni sono personaggi della mitologia greca, alcuni dei quali eseguivano compiti di cui gli dei non potevano occuparsi".

I demoni sono delle applicazioni o dei servizi che girano a lungo sul sistema e hanno molteplici compiti.

Init deve essere affidabile perchè: Gira in modalità super utente,
controlla servizi di sistema critici, e se init "muore" kernel panic.

Linea evolutiva sistemi di avvio:

1)sistemi di avvio sequenziali:

-ogni demone è avviato in sequenza,
-non esiste parallelismo.

2)sistemi di avvio gerarchici:

-forniscono funzionalità di avvio sequenziali,
-forniscono parallelismo,
-avviano servizi indipendenti in parallelo.

3)sistemi di avvio Startless/Event based:

-forniscono funzionalità di avvio gerarchico,
-On-Demand Loading (caricati su esplicita richiesta).

System-V init cos'è?

-é un demone di init,
-sistema standardizzato LSB,
-avvia/ferma i demoni sequenzialmente,
-crea i processi leggendo da /etc/inittab.

Sysvinit è l'implementazione più famosa di System-V init. e contiene anche:

-reboot serve per riavviare
-shutdown // // spegnere la macchina
-killall // // terminare i processi
-poweroff // // spegnere la macchina
-tellinit // // cambiare runlevel
-sulogin // // fare il login
-wall // // mandare un warning a tutte le tty (terminali grezzi).

LaunchD è un framework open-source per gestire avvio/spegnimento (tramite eventi) di:

-demoni
-applicazioni
-processi
-script

scritto da Dave Zarzycki (Apple inc) licenza Apache (inizialmente Apple Public License).

LaunchD sostituisce:

-init
-rc
-init.d script
-SystemStarter (Mac OS x)
-inetd / xinetd (non molto usati sono dei demoni che servono a far partire delle applicazioni quando arriva una connessione su una determinata porta.

-crond / atd (sono semplici servizi che eseguono delle opearzioni ripetutamente).

Upstart cos'è:

-sostituisce init
-event-based
-Sviluppato da Ubuntu
-adottato dalla maggior parte delle distribuzioni.

le caratteristiche di Upstart sono:

-Task/Servizi avviati/fermati tramite eventi
-eventi generati mentre Task/Servizi avviati/fermati
-eventi mandati in broadcast
-sistema di respawning dei servizi morti
(respawning. In sostanza, se il processo muore, un altro inizierà al suo posto)

-comunicazione con Upstart attraverso D-BUS

SystemD che cos'è:

-service manager per linux
-compatibile con SysV e LSB script
-ingenierizzato da Lennart Poennering (Red Hat)

e fornisce:

-parallelizzazione aggressiva
-avvio servizi tramite Socket/D-Bus
-avvio on-demand
-traccia i processi con CGROUPs
-mantiene i punti Mount/Automount

SystemV init

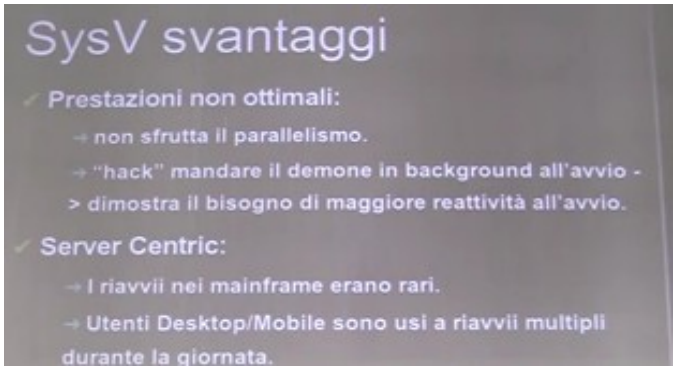
-lanciato init SO pronto
-genera i processi necessari al sistema
-legge le configurazioni da /etc/inittab

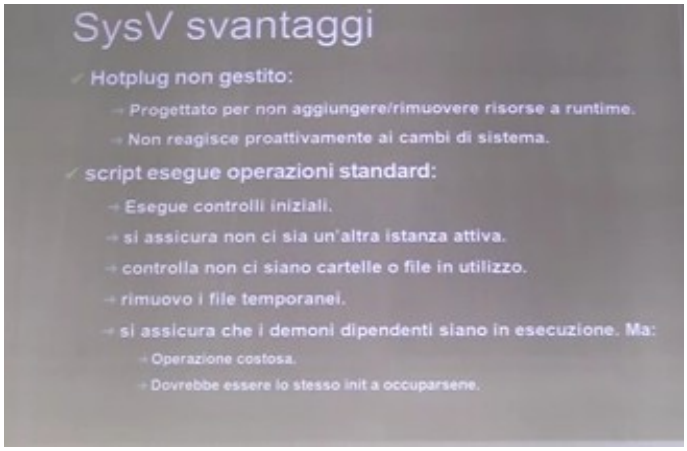
Runlevel e init
esistono 6 livelli di funzionamento:

- runlevel 0 (speciale): sistema completamente fermo
-runlevel 1 o S modalità mono utente (single user mode)
-runlevel da 2 a 5 modalità multiutente
-runlevel 6 riavvio

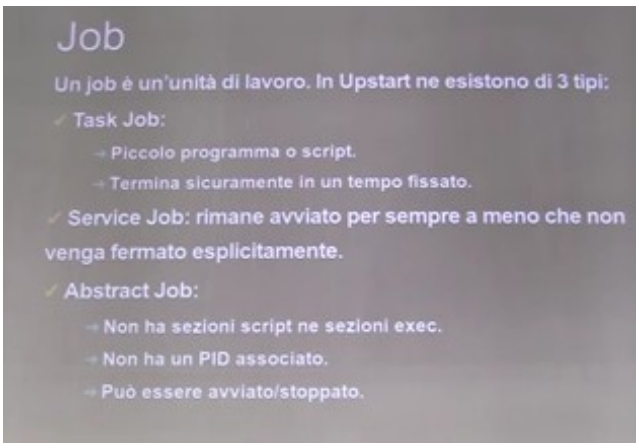
/etc/initab:

-indica ad init di lanciare alcuni processi
-definisce cosa fa ogni runlevel.





Upstart



Init e runlevel approfondimento

Init è il padre di tutti i processi, il suo ruolo principale consiste nel gestire il lancio di tutti i programmi necessari per rendere il sistema attivo creare i processi a partire da uno script in **/etc/inittab**.
Nell'inittab vengono definite le directory con gli script di avvio per i diversi runlevel (stati del sistema, in cui possono girare determinati programmi), il runlevel di default, altri script e comandi che vengono eseguiti al boot o in condizioni particolari.

Il primo script lanciato da inittab è (su RedHat) /etc/rc.d/rc.sysinit: che esegue varie operazioni tra cui:
Impostazioni di alcuni path generali nella variabile $PATH; Configurazione dell'ambiente di rete; Avvio swap per la memoria virtuale; Impostazione del nome dell'host; Check del filesystem root; Check delle quote di spazio assegnate agli utenti, se previste; Mount del filesystem root in modalità scrittura/lettura; Preparazione del sistema per caricamento dei moduli; Check delle dipendenze dei moduli; Check di tutti i filesystem ed eventuali riparazioni; Mount di tutti i filesystem; Pulizia di file di supporto al boot e di processi non più attivi; Umount dell'initrd; Impostazione dell'orologio; Attivazione dello swapping; Inizializzazione delle porte seriali; Caricamento Moduli; Attivazione dei servizi del runlevel.
Durante la fase di file system check, se si trovano errori che non possono essere riparati automaticamente è possibile che il processo di boot si blocchi. In questo caso viene richiesta la password di root e si può da bash fare un file system check manuale. Verificare su quale partizione il kernel si è fermato e scrivere (si considera che la partizione sia /dev/hda5 e abbia ext2 come file system): fsck.ext2 /dev/hda5.
Rispondere SI a tutte le lugubri domande sulla correzione di file danneggiati.

Runlevels

Nel mondo Unix ci sono 2 principali approcci al processo di startup del sistema: quello usato da Unix System V e quello, meno complesso ma meno flessibile, usato dai BSD Unix.
Linux utilizza il primo metodo, che si basa su differenti runlevel, astrazioni software per indicare diversi stati della macchina in cui possono girare diversi programmi.
In genere su Linux sono utilizzati i seguenti livelli:

Runlevel 0 : /etc/rc.d/rc0.d	Questo runlevel avvia la sequenza di arresto del sistema (shutdown)
Runlevel 1: /etc/rc.d/rc1.d	Questo runlevel rappresenta la modalità singolo utente, nessun altro utente può collegarsi, il servizio di rete è disabilitato.

Runlevel 2: /etc/rc.d/rc2.d	Rappresenta lo stato multiutente, il servizio rete è attivo ma è disabilitato il file sharing.
Runlevel 3: /etc/rc.d/rc3.d	In genere è quello predefinito quando si opera in modalità testuale, tutti i servizi sono attivi.
Runlevel 4: /etc/rc.d/rc4.d	Inutilizzato. Può essere dedicato ad usi personali
Runlevel 5: /etc/rc.d/rc5.d	E' il runlevel predefinito quando si vuole avviare Linux in modalità grafica
Runlevel 6: /etc/rc.d/rc6.d	Il runlevel 6 è quello di reboot.

Lo script /etc/rc.d/rc gestisce quali processi far partire a seconda del runlevel, andando ad analizzare le singole directory /etc/rc.d/rc#.d. In queste directory esistono una serie di symlink con nomi del tipo S12syslog o K65identd che puntano a degli script con nomi tipo /etc/rc.d/init.d/syslog o /etc/rc.d/init.d/identd.
/etc/rc.d/rc a seconda della directory corrispondente al runlevel da caricare fa partire tutti gli script che iniziano con **S** e fa chiudere tutti quelli che iniziano con **K**, eseguendoli nell'ordine indicato dal numero presente nei nomi dei file.
Gli script che di fatto permettono di gestire l'avvio o lo stop di un servizio sono quindi nella directory /etc/rc.d/init.d/ e possono essere utilizzati direttamente dall'utente per gestire i singoli processi.
Per esempio: /etc/rc.d/init.d/httpd start fa partire il server Web e /etc/rc.d/init.d/stop lo fa stoppare.
Se abbiamo il file (link a ../init/httpd) /etc/rc.d/rc3.d/S85httpd, quindi, avremo un server web avviato quando la macchina è al run-level3 (runlevel di default per un server, che non ha bisogno di Xwindows).
Se vogliamo evitare che venga avviato un server web, bastera rinominare il file, sostituendo la K alla S:
mv /etc/rc.d/rc3.d/S85httpd /etc/rc.d/rc3.d/K85httpd
Nel fare queste operazioni va sempre considerato il numero dopo la prima lettera, che determina l'ordine di esecuzione degli script.

Gestione dei servizi al boot

La soluzione sopra esposta per gestire quali servizi deve offrire il nostro server è effettivamente poco immediata (ma, come si è visto, piuttosto flessibile). Esistono tool che permettono di gestire più facilmente quali servizi avviare al boot.
Per esempio, su distribuzioni RedHat, utilizzando il comando **ntsysv** è possibile accedere ad un tool grafico (su interfaccia testuale) dedicato.
I processi identificati da un star (*) e dal nome del servizio indicano che il processo verrà attivato al boot, altrimenti i processi identificati solo dal nome del servizio e da una casella vuota indicano i processi che non verranno fatti partire.
Alternativamente si può usare il comando **serviceconf** su interfaccia grafica o il comando shell **chkconfig**.
E' molto importante disattivare tutti i processi e servizi inutili per evitare sprechi delle risorse del sistema.

Gestione e creazione di servizi in Debian

Introduzione

La maggior parte dei servizi installati su un sistema Debian GNU/Linux storicamente viene avviata e fermata tramite un apposito script che si trovava sotto la directory `/etc/init.d/`, tramite il sistema di init SysV. A partire da Debian 8 (Jessie) questo sistema è stato sostituito di default da systemd, che ha una nuova sintassi e nuovi percorsi per i propri file di configurazione, e non richiede più nemmeno l'impiego di script.

Systemd è tuttavia in grado di comprenderne la sintassi, purché non ci siano servizi di systemd con lo stesso nome che nasconderebbero gli script presenti in `/etc/init.d/`, e di eseguirli all'avvio come avveniva in precedenza, secondo le dipendenze indicate in forma di commento nell'intestazione dello script.

È possibile inoltre utilizzare il comando `service`, che si occuperà di invocare `systemctl` se systemd è attivo oppure di invocare lo script in `/etc/init.d/` in un ambiente pulito.

Ad esempio, per avviare il servizio MySQL, è sufficiente con privilegi di amministrazione il seguente comando:

```
# service mysql start
```

e per fermare il servizio:

```
# service mysql stop
```

Come si creano gli script di avvio

Suggerimento
Se si è interessati soltanto a eseguire dei comandi dopo che il sistema e i servizi sono stati avviati, senza lanciare qualcosa che resti in esecuzione, è sufficiente modificare il file `/etc/rc.local`, aggiungendo lì i propri comandi.

Per creare uno script di avvio per un servizio, è sufficiente creare un nuovo file sotto la directory `/etc/init.d/` e poi editarlo con un qualsiasi editor di testi.

A partire da Debian 8 (Jessie), se si utilizza systemd, è necessario accertarsi prima che il nome non sia già utilizzato da un servizio di systemd, altrimenti sarà ignorato:

```
$ systemctl status nome-script-da-creare
```

Se il nome è libero questo comando restituirà un messaggio di errore che il file non esiste.

Ogni script di avvio che si rispetti ha almeno una sezione nella quale controlla i parametri che gli sono stati passati e altre in cui si occupa poi di eseguire un diverso comando a seconda del parametro passato.

Come per ogni script, anche per questi bisogna indicare nella prima riga l'interprete che deve essere utilizzato per eseguirli, ad esempio:

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

La selezione del comando da eseguire viene fatta attraverso un semplice `case/esac` di Bash. Un esempio di script è riportato nel listato seguente, che in questo caso è stato scritto nel file `/etc/init.d/mio_start_script.sh`

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          mio_start_script.sh
# Required-Start:    hal
# Required-Stop:
# Should-Start:
# Should-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Attivazione ottimizzazioni Notebook
# Description:       Attivazione ottimizzazioni Notebook
### END INIT INFO

case "$1" in
start)  echo "Attivo ottimizzazioni Notebook:"

        echo -e "\n1) abilito risparmio energetico per ac97"
        echo 1 > /sys/module/snd_ac97_codec/parameters/power_save

        echo -e "\n\n2) controllo se hdparm è installato correttamente"
        test -f /sbin/hdparm || exit 0

        echo -e "\n2.1) abilito il MultiCount16"
        hdparm -m16 /dev/hde
        echo -e "\n2.2) abilito l'accesso a 32 bit"
        hdparm -c3 /dev/hde
        echo -e "\n2.3) abilito il buffer a 2048"
        hdparm -a2048 /dev/hde
        ;;
stop)   echo "Non ancora implementato"
        ;;
restart) echo "Non ancora implementato"
        ;;
reload|force-reload) echo "Non ancora implementato"
        ;;
*)      echo "Usage: /etc/init.d/mio_start_script.sh
{start|stop|restart|reload|force-reload}"
        exit 2
        ;;
esac
exit 0
```

Lo script andrà poi reso eseguibile con:

```
# chmod +x /etc/init.d/mio_start_script.sh
```

systemctl

Con systemd è sufficiente abilitare lo script:

```
# systemctl enable mio_start_script.sh
```

`update-rc.d`

Una volta creato lo script bisogna renderlo automatico. Per automatizzare lo start e lo stop di un servizio si possono scegliere 2 strade:

1. si creano i vari link simbolici per stoppare e far partire nelle rispettive directory `/etc/rcN.d`
2. si utilizza il più comodo comando `update-rc.d`, che è la forma consigliata (in caso si utilizzi `systemctl`, verrà sempre invocato questo comando)

Il comando da dare è quindi:

```
# update-rc.d <nome_script_in_init.d> default <priorità_start> <priorità_stop>
```

dove:

- `<nome_script_in_init.d>` nel nostro caso è `mio_script_start.sh`;
- `<priorità_start>` il numero di sequenza che decide l'ordine con cui eseguire lo script quando va avviato;
- `<priorità_stop>` il numero di sequenza che decide l'ordine con cui eseguire lo script quando va stoppato.

Ad esempio:

```
# update-rc.d mio_script_start.sh 10 50
```

restituisce come output:

```
Adding system startup for /etc/init.d/mio_script_start.sh ...
/etc/rc0.d/K50mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc1.d/K50mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc6.d/K50mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc2.d/S10mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc3.d/S10mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc4.d/S10mio_script_start.sh -> ../init.d/mio_script_start.sh
/etc/rc5.d/S10mio_script_start.sh -> ../init.d/mio_script_start.sh
```

Se non volessimo addentrarci nell'argomento priorità di avvio, il mio consiglio è di usare l'opzione `defaults`; in questo modo sarà la nostra Debian ad assicurarsi che il servizio venga installato nei vari slot di priorità in maniera automatica, in base a quanto configurato nell'intestazione dello script.

```
# update-rc.d mio_script_start.sh defaults
```

Eliminazione di un servizio

Nel caso in cui il nostro servizio `mio_script_start.sh` non dovesse più esserci utile, dobbiamo disabilitarlo.

Se si utilizza systemd è meglio prima disabilitare il servizio, mentre non è necessario altrimenti:

```
# systemctl disable mio_script_start.sh
```

Sia con systemd sia senza, per eliminare i relativi link simbolici (che non avrebbero più effetto)

creati nelle cartelle `/etc/rc?.d/`, si può usare il comando:

```
# update-rc.d -f mio_script_start.sh remove
```

Infine possiamo rimuovere lo script dalla directory `/etc/init.d` con:

```
# rm /etc/init.d/mio_script_start.sh
```