Upstart e Systemd

(sull´immagine di apertura: bootstrap) In ogni classico sistema operativo Unix-derivato, una volta espletati i compiti di inizializzazione dell'hardware, di montaggio dei device e via discorrendo, il kernel lancia, quale primo processo, **init** (/sbin/init), che si occupa in user space di lanciare i servizi configurati per il **runlevel** (possibile stato del sistema) in cui il sistema andrà a porsi. A seconda del runlevel il sistema compie determinate azioni.

Debian e Ubuntu, ad esempio, usano i seguenti runlevel:

- 1: modalità a singolo utente, con la directory radice / montata in sola lettura, per compiti di troubleshooting;
- 2 a 5: modalità multiutente: sono di fatto equivalenti;
- 0: arresta il sistema, = halt;
- 6: riavvia il sistema, = reboot.

Il procedimento di avvio viene controllato dal file **/etc/inittab**, il quale fa in modo che avvenga quanto segue.

Quando si entra in un runlevel tutti i **file in /etc/rc[runlevel].d/** vengono eseguiti. La prima lettera del nome determina il modo in cui lo script viene lanciato: quelli che iniziano con K (kill) vengono lanciati con l'argomento stop (lo script deve prevederlo). Quelli che iniziano per S (start) vengono lanciati con l'argomento start. I file vengono **eseguiti in ordine alfabetico** (i servizi gireranno quindi in background, "staccandosi" da init); per cui quelli stop vengono lanciati prima di quelli start e i numeri a due cifre che seguono K o S determinano l'ordine in cui vengono eseguiti.

I file in /etc/rc[runlevel].d sono semplici **collegamenti simbolici** agli **script in /etc/init.d/**: qui saranno salvati i veri e propri script di gestione (lancio, arresto, riavvio) dei servizi.

Quindi, per **avviare/fermare un servizio** (ad opera di root) è semplicemente necessario, rispettivamente:

```
/etc/init.d/servizio start
/etc/init.d/servizio stop
```

Su Debian vengono eseguiti anche gli script in **rcS.d/**, indipendentemente dal runlevel, e lo script **rc.local** come ultimo script per tutti i runlevel i multiutente.

È possibile porre in esecuzione automatica all´avvio un servizio mediante (Debian/Ubuntu):

`update-rc.d servizio defaults`: utilizzando le informazioni presenti nello script stesso, lo inserirà sui runlevel scelti; se le informazioni non sono presenti, il comando lo installa in avvio sui runlevel 2, 3, 4 e 5, ed in arresto su 0, 1 e 6;

`update-rc.d -f servizio remove`: rimuove da tutti i runlevel ogni link allo script (che quindi non verrà più lanciato all´avvio del sistema)

La procedura di attivazione di un servizio all´avvio viene automaticamente impostata durante l

´installazione del demone tramite apt-get.

LIMITAZIONI

Il comportamento visto è relativo al **“SysV init” classico**, ma, non da molto, le distribuzioni hanno iniziato ad introdurre variazioni sul tema.

La problematica più importante circa l’init classico risiede nella sua **sequenzialità**: gli script di avvio di servizi o programmi vengono lanciati sequenzialmente e se uno di essi blocca i rimanenti (per un determinato tempo massimo, mai indefinitamente), gli altri altro non fanno che aspettare.

DEPENDENCY-BASED BOOT

Debian 6 ha introdotto il concetto di **dependency-based boot**, nel quale il sostituto dell´init classico si occupa essenzialmente (e non si spinge oltre) di **parallelizzare l’esecuzione** degli script.

Per fare ciò, ogni vecchio script di avvio dev´esser parzialmente modificato aggiungendo le informazioni circa le sue dipendenze di run-time, come esemplificato di seguito per il demone exim4 su Debian Squeeze:

```
### BEGIN INIT INFO
# Provides: exim4
# Required-Start: $remote_fs $syslog $named $network $time
# Required-Stop: $remote_fs $syslog $named $network
# Should-Start: postgresql mysql clamav-daemon greylist spamassassin
# Should-Stop: postgresql mysql clamav-daemon greylist spamassassin
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: exim Mail Transport Agent
# Description: exim is a Mail Transport agent
### END INIT INFO
```

I campi più importanti per la definizione delle dipendenze sono chiaramente **Required-Start**, che definisce quali risorse debbano esser presenti per poter avviare il presente script, **Should-Start**, che definisce quali servizi, se presenti, debbano essere avviati prima del presente e **Should-Stop** il quale dà la lista di quali servizi, se presenti, dovrebbero esser terminati dopo la terminazione del presente.

Come ci si aspetta, **Default-Start** e **Default-Stop** definiscono rispettivamente in quali runlevel avviare e fermare il servizio facente capo a questo script di avvio.

Lo script eseguibile exim4 posto in /etc/init.d viene inserito nella lista degli script avviabili (cioè ne vengono creati i collegamenti come sopra spiegato) mediante il comando:

```
insserv exim4
```

sebbene il "vecchio" metodo funzioni ancora:

```
update-rc.d exim4 defaults
```

UPSTART

Quale sostituto di init, Ubuntu utilizza **Upstart**, che mira all'accelerazione del processo di boot utilizzando una logica diversa.

Upstart è **orientato agli eventi** e lavora utilizzando il concetto di job. I job file sono gli script di /etc/init, ma non c'è una sequenza specifica: ogni job specifica gli eventi ai quali fare da handler e, quando questi eventi occorrono, Upstart lancia tutti gli handler **in parallelo**.

Un evento basilare è chiaramente rappresentato da startup (lanciato dallo stesso Upstart all'avvio): tutti i job che devono essere lanciati allo startup conterranno il codice:

```
start on startup
```

Altro chiaro evento basilare è quello legato allo stop dei servizi.

Upstart è più flessibile dell'init classico (e del dependency-based boot init), in quanto lancia i servizi **solo se effettivamente servono**. Immaginiamo che un computer venga avviato senza scheda di rete: se init avvia comunque ogni servizio relativo alla rete, Upstart non lo fa fino a che non venga lanciato l'evento network up. Stessa cosa vale per un sistema di stampa: a meno che la macchina non sia fisicamente collegata a una stampante, o una applicazione richiede di stampare qualcosa, non è necessario eseguire un demone di stampa come CUPS.

Per compatibilità col passato, Upstart è anche in grado di lanciare i canonici script di init.d/ non modificati. Quando tutti gli script di avvio saranno aggiornati, Ubuntu non contemplerà più il concetto di runlevel.

SYSTEMD

Fedora 15 utilizza invece systemd, sistema che presto verrà adottato da alcune altre distribuzioni, anch´esso capace di **parallelizzazione** nel lancio dei servizi (più spinta che Upstart) e di **avvio on-demand**.

Come riporta la guida a systemd di Fedora, systemd, al fine di accelerare l´esecuzione in fare di boot, utilizza una nuova tecnologia chiamata **socket activation**. Tutti i processi vengono lanciati da systemd contemporaneamente ed è il sistema a gestirne quindi lo stato.

La soluzione che Lennart Poettering (il suo ideatore) ha ideato è quella di **portare fuori** dai demoni **il binding ai vari socket** demandando questo lavoro al sistema di init, che a questo punto non è solo un sistema di lancio dei processi ma si incarica di inizializzare e attivare i socket principali per il funzionamento del sistema operativo in un unico iniziale step (prima del lancio dei vari processi).

La creazione di tutti i socket in un unico punto permette di parallelizzare il successivo lancio dei processi (che devono essere modificati per permettere ciò) senza dover dar bado alle **dipendenze reciproche** nel loro avvio.