Init

il processo 1: Init

- nei sistemi operativi unix like nuovi processi possono essere generati solamente per filiazione utilizzando la system call fork (man 2 fork)
- la relazione figlio-padre stabilisce una gerarchia ad albero che unisce tra loro tutti i processi
- comandi: pstree
- il processo Init (/sbin/init) è il padre di tutti i processi
  - viene eseguito direttamente dal kernel
  - il suo identificatore di processo (PID) è 1
  - ha il compito di generare i processi necessari al funzionamento del sistema, in base a quanto specificato in /etc/inittab

/etc/inittab

- stabilisce quali processi debbano essere creati all'avvio del sistema ed al verificarsi di eventi eccezionali
- è strutturato in livelli di esecuzione (runlevel)
- ogni riga è un record a 4 campi che descrive l'esecuzione di un processo

id **:** ***livelli*** **:** ***azione*** **:** ***processo***

  - id identificatore di record
  - livelli lista di runlevel ai quali si applica il record
  - azione azione da eseguire
  - processo processo da eseguire

/etc/inittab

- le azioni più comuni
  - initdefault (runlevel di default)
  - sysinit (esegui il processo al boot del sistema, ignora runlevel)
  - wait (esegui il processo ed attendi che termini)
  - respawn (riesegui il processo quando termina)
  - ctrlaltdel (esegui il processo quando init riceve SIGINT)
  - power{wait,failnow,okwait} (esegui il processo quanto init riceve SIGPWR)
  - kbrequest (esegui il processo quando viene effettuata una SysReq da tastiera)
- riferimenti:
  - appunti, cap. 60
  - man 5 inittab

/etc/inittab

- esempio (Debian GNU/Linux):

```
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
~~: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
z6:6:respawn:/sbin/sulogin
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
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
```

Runlevel

- ad ogni runlevel è idealmente associato un insieme di servizi attivi quando init si trova in quel runlevel
- a runtime è possibile indicare ad init di cambiare runlevel
  - al cambio di runlevel è necessario cambiare l'insieme dei servizi in esecuzione in modo che corrispondano al nuovo runlevel
- /etc/init.d/rc (o analoghi) è lo script che si occupa di mantenere la consistenza tra runlevel e servizi attivi
- semantica usuale dei runlevel

0	halt
1	single-user
2-5	multi-user
6	reboot

Gestione dei servizi

- al fine di facilitare il compito di /etc/init.d/rc, ad ogni servizio corrisponde uno script responsabile di gestirne l'esecuzione
- questi script, uno per servizio, sono locati in /etc/init.d/ (o analoghi) ed implementano una interfaccia comune:
  - start attiva il servizio
  - stop disattiva il servizio
  - restart sequenza: disattiva il servizio (se attivo), attiva il servizio
  - reload ricarica la configurazione del serizio, evitando restart
  - force-reload come reload (se possibile), altrimenti restart
- e.g. /etc/init.d/cron, /etc/init.d/nethack-common
- riferimenti: Debian Policy Manual, 9.3.2

Gestione dei servizi

- l'abbinamento servizi-runlevel è stabilito dai link simbolici presenti nelle directory /etc/rc*<runlevel>*.d/ (o analoghe), una pre runlevel
- ognuna di esse contiene link aventi nomi della forma

  {S,K}*<nn><nome-servizio>*
  - e.g.: /etc/rc2.d/S89cron, /etc/rc6.d/K11cron
  - i link puntano agli script di gestione dei servizi
- la lettera iniziale stabilisce se, al momento del passaggio ad un dato runlevel il servizio debba essere attivato (lettera “S”) o disattivato (lettera “K”)
- il numero a due cifre stabilisce l'ordine di esecuzione delle attivazioni/disattivazioni

Procedura di boot System V

0. il kernel linux è installato in una locazione accedibile al boot
1. [bios] avvio della macchina, caricamento della parte monolitica del kernel in memoria
2. [kernel] esecuzione di init
3. [init] esecuzione dei processi corrispondenti ad azioni sysinit, boot e bootwait (e.g. /etc/init.d/rcS)
4. [init] selezione del runlevel di default (azione initdefault)
5. [init] esecuzione dei processi corrispondenti al runlevel selezionato (e.g. /etc/init.d/rc 2)
6. [/etc/init.d/rc] esecuzione di tutti i servizi che devono essere eseguiti nel runlevel selezionato (e.g. /etc/rc2.d/S*)

Pianificazione

Scheduling di attività

- molte delle pratiche relative all'amministrazione di sistemi GNU/Linux devono essere effetuate periodicamente
  - backup
  - raccolta/rotazione di log
  - rimozione di dati temporanei e cache (e.g. /tmp, news)
  - aggiornamento di db (e.g. locate, manpages)
- la pianificazione di tali attività viene solitamente affidata ai seguenti tool
  - cron
  - at
  - anacron

Cron

- cron è un servizio di sistema configurabile per eseguire programmi in momenti prestabiliti
- caratteristiche:
  - è un demone, solitamente in esecuzione nei runlevel multiuser
  - è configurabile per-utente: ogni utente può specificare quali programmi vuole eseguire in quali momenti, tali programmi vengono eseguiti con l'UID dell'utente
  - ha granularità di 1 minuto
  - l'eseguibile che lo implementa non fa altro che “dormire”, risvegliandosi ogni minuto per verificare se ci siano o meno programmi da eseguire

Crontab

- la configurazione per-utente di cron risiede in file denominati crontab, residenti in /var/spool/cron/, editabili utilizzando /usr/bin/crontab
  - è possibile visualizzare (crontab -l), eliminare (crontab -u) e modificare (crontab -e) il proprio crontab
- i file crontab sono file di testo, con la seguente sintassi
  - righe vuote e commentate (che inizino per “#”) sono ignorate
  - ogni altra riga rappresenta una specifica cron (o un assegnamento nel caso di Vixie Cron) composta da 6 campi:

    <min> <ora> <data> <mese> <giorno_settimana> <comando>
  - *comando* è un comando che verrà eseguito dalla shell predefinita (variabile d'ambiente SHELL)

Crontab

  - ogni altro campo può avere la forma
    - numero naturale (e.g. 6)
    - wildcard (e.g. *)
    - intervallo (e.g. 1-4)
    - elenco (e.g. 1,3,4)
    - passo (e.g. */10)
- il comando viene eseguito quando l'orologio di sistema corrisponde, componente per componente, ai primi 5 campi della specifica cron, secondo semplici regole di *matching*
- riferimenti: man 8 cron, man 5 crontab, appunti cap.64
- comandi: crontab

Crontab di sistema

- il file /etc/crontab rappresenta il crontab di sistema
  - è modificabile manualmente
  - contiene specifiche cron *estese*, dotate di un campo aggiuntivo *<utente>* prima del comando che specifica lo username il cui UID verrà utilizzato per eseguire il comando
- estensioni (Debian e altre distribuzioni)
  - /etc/cron.d/* specifiche cron estese, una per file
  - /etc/cron.{hourly,daily,weekly,monthly}
    script da eseguire su base oraria/giornaliera/..., solitamente invocati da /etc/crontab o /etc/cron.d/
  - specifica temporale “@reboot” in /etc/crontab, per indicare comandi pianificati al boot

Anacron

- cron “non ha memoria”
  - se non si sveglia nel momento in cui un comando è pianificato (e.g. la macchina è spenta in quell'istante, o il servizio non è attivo), il comando non verrà eseguito fino alla pianificazione successiva (se esiste!)
- anacron è un sistema di pianificazione dotato di memoria
  - pensato per macchine non sempre accese
  - granularità giornaliera
  - configurazione system-wide
- riferimenti:
  - man 8 anacron, man 5 anacrontab
  - http://anacron.sourceforge.net/

Anacrontab

- la configurazione di anacron risiede in /etc/anacrontab
  - file testuale, ogni riga può essere o un assegnamento di variabili o una specifica anacron
  - una specifica cron è composta da 4 campi:

    <cadenza> <delay> <job-name> <comando>

    - cadenza rappresenta la periodicità (in giorni) dell'esecuzione
    - delay rappresenta il ritardo (in minuti) prima dell'esecuzione
      - al fine di evitare esecuzione contemporanea dei comandi, che potrebbe caricare troppo la macchina
    - job-name è il nome simbolico del job
    - comando è il comando da eseguire

Esecuzione di anacron

- anacron viene solitamente eseguito al boot della macchina e periodicamente via cron
- per ognuno dei job specificati nell'anacrontab
  - controlla che non sia passato un numero di giorni maggiore alla sua cadenza
    - le informazioni (timestamp) sulle passate esecuzioni sono memorizzate in /var/spool/anacron/
  - se tale tempo è trascorso (o se il job non è mai stati eseguito)
    - lo esegue attendendo un numero di minuti pari al delay
    - salva in /var/spool/anacron/ il timestamp relativo a questa esecuzione del job

At: cron one-shot

- il demone atd è simile a cron, ma pensato per eseguire comandi una sola volta, non appena viene raggiunto un dato momento
  - dispone di una coda di eventi, configurabile utilizzando i comandi: “at” (aggiunge un evento alla coda), “atq” (mostra la coda), “atrm” (rimuove un evento dalla coda)
  - offre una sintassi più semplice di quella dei crontab, e.g.
    - at 15:30 + 2 days
    - at 8:00 tomorrow
    - at 10:00 10.11.2005
- comandi: at, atq, atrm

Pacchetti

Applicativi distribuiti in forma sorgente

- le licenze di applicativi software libero garantiscono l'accesso ai sorgenti
  - conseguenza pratica sulla distribuzione: la totalità di tali applicativi sono distribuiti dagli autori in forma sorgente
- procedura tipica di installazione applicativi:
  1. donwload dell'archivio dei sorgenti [client http/ftp/...]
  2. estrazione dei sorgenti [tar]
  3. configurazione dei sorgenti [configure]
     - verifica automatica delle dipendenze [configure]
     - risoluzione manuale dei problemi derivanti [amministratore]
  4. compilazione da sorgenti a binari [make]
  5. installazione dei binari [make]

Applicativi distribuiti in forma sorgente

- vantaggi
  - uniformità della procedura di installazione (a volte)
  - incoraggia portabilità e studio dei sorgenti
  - flessibilità nella scelta delle opzioni di configurazione
- svantaggi
  - lunghi tempi di compilazione
  - necessità di applicativi e librerie a compile-time, non necessari a run-time
  - inefficienza nell'uso di risorse di calcolo comunitarie (sarebbe sufficiente una compilazione per ogni architettura)
  - risoluzione manuale delle dipendenze
  - gestione degli upgrade
  - (configurazione manuale dei sorgenti)

Applicativi distribuiti in forma binaria

- vantaggi
  - duali al caso degli applicativi distribuiti in forma sorgente
- svantaggi
  - dipendenza dall'architettura
  - dipendenza da librerie dinamiche
  - fiducia in chi ci fornisce il binario

Pacchetti applicativi

- per superare gli svantaggi della distribuzione degli applicativi, ogni distribuzione GNU/Linux offre l'astrazione del “pacchetto”
  - ogni pacchetto rappresenta una componente del sistema
    - sono componenti: applicativi binari, librerie, documentazione, dati, sorgenti, ...
  - ogni pacchetto è associato ad una versione
    - esiste un ordinamento totale tra versioni
  - i pacchetti definiscono la granularità alla quale è possibile aggiungere e rimuovere componenti del sistema
  - esistono relazioni binarie che legano tra loro i pacchetti
    - e.g.: “dipende da”, “è incompatibile con”
- ogni distribuzione offre il proprio sistema di gestione dei pacchetti

Vita di un pacchetto

- esistono due tipi di pacchetti: sorgenti e binari
  - i pacchetti binari sono composti da ciò che è necessario per l'utilizzo a runtime di una componente di sistema
    - e.g. eseguibili binari, librerie compilate
  - i pacchetti sorgenti contengono tutte le informazioni necessarie per generare i pacchetti binari corrispondenti
    - e.g. forma sorgente di un applicativo o di una libreria
- la vita di un pacchetto è articolata nelle seguenti fasi:
  1. rilascio della versione X di una componente
  2. creazione della versione X del pacchetto sorgente
  3. compilazione delle versioni X dei pacchetti binari (un pacchetto binario per ogni architettura supportata)

Utilizzo di un pacchetto

- l'amministratore di sistema non deve necessariamente interagire con la vita di un pacchetto
- ogni sistema di gestione dei pacchetti
  - presenta all'amministratore:
    - l'insieme dei pacchetti (binari) installati
    - un insieme di pacchetti (binari) disponibili
  - permette di eseguire azioni sui pacchetti:
    - installare un pacchetto (binario) non installato
    - disinstallare un pacchetto (binario) installato
    - aggiornare un pacchetto (binario) installato all'ultima versione
  - assicura la consistenza delle relazioni tra pacchetti (binari)
    - e.g. impedisce di installare un pacchetto (binario) senza installare i pacchetti da cui esso dipende

Pacchetti applicativi

- vantaggi
  - semplicità e velocità di gestione delle componenti di sistema
  - non richiede l'installazione di applicativi e librerie necessari solo compile-time
  - ogni pacchetto sorgente viene compilato una sola volta (per architettura)
- svantaggi
  - fiducia nei manutentori della distribuzione
  - rigidità nella scelta delle opzioni di configurazione
- riferimenti: appunti, capp.38-39

Case study: distribuzione Debian

- sorgenti vs binari
- dipendenze
  - depends, recommends, suggests, conflicts, replaces, provides
- stato dei pacchetti
  - installed, half-installed, not-installed, unpacked, half-configured, config-files
- azioni sui pacchetti
  - install, remove, purge
- gestione dei pacchetti, la gerarchia Debian:
  - dpkg-deb / dpkg / APT / {dselect,aptitude,synaptic}

Case study: distribuzione Debian

- comandi: dpkg, apt-get, apt-cache
- riferimenti
  - appunti, cap. 42–46
  - Debian Tutorial
    http://www.debian.org/doc/manuals/debian-tutorial/
  - APT HOWTO http://www.debian.org/doc/manuals/apt-howto/