Package Manager e sistemi di init

Simone Lombardi

Università degli studi di Salerno

smlb@archlinux.info
http://smlb.me

24 Ottobre 2014

Overview

1. Prima Parte
   - Package Manager
2. Seconda Parte
   - Sistemi di init

Introduzione sui package manager

Un package manager comprende una serie di strumenti utilizzati per gestire in modo automatico ed intuitivo il software in una distribuzione GNU/Linux. Possiamo installare, rimuovere ed aggiornare i package (il metodo differisce da distro a distro).

Funzioni dei package manager

- gestione delle dipendenze (non tutti hanno questa funzione)

Funzioni dei package manager

- gestione delle dipendenze (non tutti hanno questa funzione)
- verifica del checksum di ogni singolo pacchetto da installare, ovvero il controllo della correttezza e completezza

Funzioni dei package manager

- gestione delle dipendenze (non tutti hanno questa funzione)
- verifica del checksum di ogni singolo pacchetto da installare, ovvero il controllo della correttezza e completezza
- verifica di eventuali firme digitali, ovvero controllo della chiave GPG per verificare l'attendibilità di un package

Funzioni dei package manager

- gestione delle dipendenze (non tutti hanno questa funzione)
- verifica del checksum di ogni singolo pacchetto da installare, ovvero il controllo della correttezza e completezza
- verifica di eventuali firme digitali, ovvero controllo della chiave GPG per verificare l'attendibilità di un package
- gestione di operazioni sui pacchetti installati, come aggiornamenti e in casi particolari anche downgrade (ritornare a versione precedente)

Suddivisione package manager

- **abs, pacman**: Arch Linux

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate
- **PackageKit**: cross-distro

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate
- **PackageKit**: cross-distro
- **pkgtool**: Slackware

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate
- **PackageKit**: cross-distro
- **pkgtool**: Slackware
- **xbps**: Void Linux

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate
- **PackageKit**: cross-distro
- **pkgtool**: Slackware
- **xbps**: Void Linux
- **yum, dnf, rpm**: Fedora e derivate

Suddivisione package manager

- **abs, pacman**: Arch Linux
- **apt, aptitude, dpkg**: Debian e derivate
- **emerge, portage**: Gentoo e derivate
- **PackageKit**: cross-distro
- **pkgtool**: Slackware
- **xbps**: Void Linux
- **yum, dnf, rpm**: Fedora e derivate
- **zypper**: OpenSUSE

Differenze fra package manager

Nome	Risolve dipendenze	Formato
apt	✓	.deb
emerge	✓	ebuilds
dpkg	NO	.deb
pacman	✓	.pkg.tar.xz
yum	✓	.rpm
pkgtool	NO	.txz
zypper	✓	.tgz

Table: Confronto dei vari package manager

Introduzione all'init

Nei sistemi UNIX, un init è il primo processo avviato durante il boot ed è un demone che verrà killato quando viene effettivamente spenta la macchina. L'init è avviato dal kernel: ha PID 1 e si occupa di gestire varie operazioni all'interno di un calcolatore:

Introduzione all'init

Nei sistemi UNIX, un init è il primo processo avviato durante il boot ed è un demone che verrà killato quando viene effettivamente spenta la macchina. L'init è avviato dal kernel: ha PID 1 e si occupa di gestire varie operazioni all'interno di un calcolatore:

- Avviare demoni (e/o script dell'utente)

Introduzione all'init

Nei sistemi UNIX, un init è il primo processo avviato durante il boot ed è un demone che verrà killato quando viene effettivamente spenta la macchina. L'init è avviato dal kernel: ha PID 1 e si occupa di gestire varie operazioni all'interno di un calcolatore:

- Avviare demoni (e/o script dell'utente)
- Gestire/Verificare il runlevel di una macchina

Introduzione all'init

Nei sistemi UNIX, un init è il primo processo avviato durante il boot ed è un demone che verrà killato quando viene effettivamente spenta la macchina. L'init è avviato dal kernel: ha PID 1 e si occupa di gestire varie operazioni all'interno di un calcolatore:

- Avviare demoni (e/o script dell'utente)
- Gestire/Verificare il runlevel di una macchina
- Effettuare check in fase di spegnimento e accensione

Runlevel

Controllano quali processi/servizi sono avviati automaticamente dall'init, abbiamo sette runlevel:

- 0: Halt
- 1: Single-User Mode
- 2: Multi-User Mode
- 3: Multi-User con Network
- 4: Definito dall'utente
- 5: Avvia il sistema normalmente
- 6: reboot

NB: *questa è la lista dei runlevel standard, possono anche differire.*

SysVinit

- PID 1

SysVinit

- PID 1
- Lancia processi via /etc/inittab

SysVinit

- PID 1
- Lancia processi via /etc/inittab
- Script contenuti in /etc/rc.d/init.d/ o /etc/init.d/

OpenRC

- Portabile su tutti gli OS

OpenRC

- Portabile su tutti gli OS
- Codici e configurazioni separate

OpenRC

- Portabile su tutti gli OS
- Codici e configurazioni separate
- Ordinamento dell'avvio dei demoni automatico

OpenRC

- Portabile su tutti gli OS
- Codici e configurazioni separate
- Ordinamento dell'avvio dei demoni automatico
- Controllo e gestione via *rc*, *rc-update* e *rc-status*.

Upstart

- Scritto usando libnih (equivalente di altre lib C come glib)

Upstart

- Scritto usando libnih (equivalente di altre lib C come glib)
- Avviato da super-user

Upstart

- Scritto usando libnih (equivalente di altre lib C come glib)
- Avviato da super-user
- Gestisce i servizi *critici* del sistema

Upstart

- Scritto usando libnih (equivalente di altre lib C come glib)
- Avviato da super-user
- Gestisce i servizi *critici* del sistema
- Se l'init *muore*, c'è un kernel panic

Systemd

- Parallelizzazione

Systemd

- Parallelizzazione
- Processi: avviati via socket e attivazione via D-Bus

Systemd

- Parallelizzazione
- Processi: avviati via socket e attivazione via D-Bus
- Unit Files: gestione di varie procedure tramite files con estensioni diverse (.service, .timer, .socket, .mount etc)

Systemd

- Parallelizzazione
- Processi: avviati via socket e attivazione via D-Bus
- Unit Files: gestione di varie procedure tramite files con estensioni diverse (.service, .timer, .socket, .mount etc)
- Journaling: in formato binario, interrogabile con *journalctl*

Systemd

- Parallelizzazione
- Processi: avviati via socket e attivazione via D-Bus
- Unit Files: gestione di varie procedure tramite files con estensioni diverse (.service, .timer, .socket, .mount etc)
- Journaling: in formato binario, interrogabile con *journalctl*
- Compatibilità: GNU/Linux-only

Systemd

- Parallelizzazione
- Processi: avviati via socket e attivazione via D-Bus
- Unit Files: gestione di varie procedure tramite files con estensioni diverse (.service, .timer, .socket, .mount etc)
- Journaling: in formato binario, interrogabile con *journalctl*
- Compatibilità: GNU/Linux-only
- Troppe *features* da inserire in una lista

Altri sistemi di init

- **Runit**: default in Void Linux
- **Sinit**: sviluppato dal team s3r0 (suckless)
- **minirc**: minimalistic-init-system

README

Queste slides sono realizzate con LaTeXe rilasciate sotto GFDL. I sorgenti risiedono su Github: sono ottenibili e modificabili liberamente.

runlevel 0
The End