

***Pacchetti e
gestione del
software in
ambiente Linux***

Layout

- Cosa serve
 - Aver installato una distribuzione
 - Essere interessati ai pacchetti :)
- Cosa vedremo
 - Tipi di pacchetti
 - Gestione dei pacchetti
 - Ottimizzazione del software
 - Approccio di alcune distribuzioni Linux

Software opensource

Spesso viene rilasciato solo come codice sorgente

Necessita di:

- Configurazioni precompilazione
- Compilazione
- Configurazioni postinstallazione

Software opensource (2)

Installazione “a mano”

- Configurazione (./configure)
- Compilazione (make)
- Installazione (make install)
- Impostazioni per il funzionamento

Ma..... è comodo questo approccio?

NO!

Software precompilato

- Il software non deve essere compilato:
 - Meno tempo di CPU
 - Si evita la configurazione precompilazione

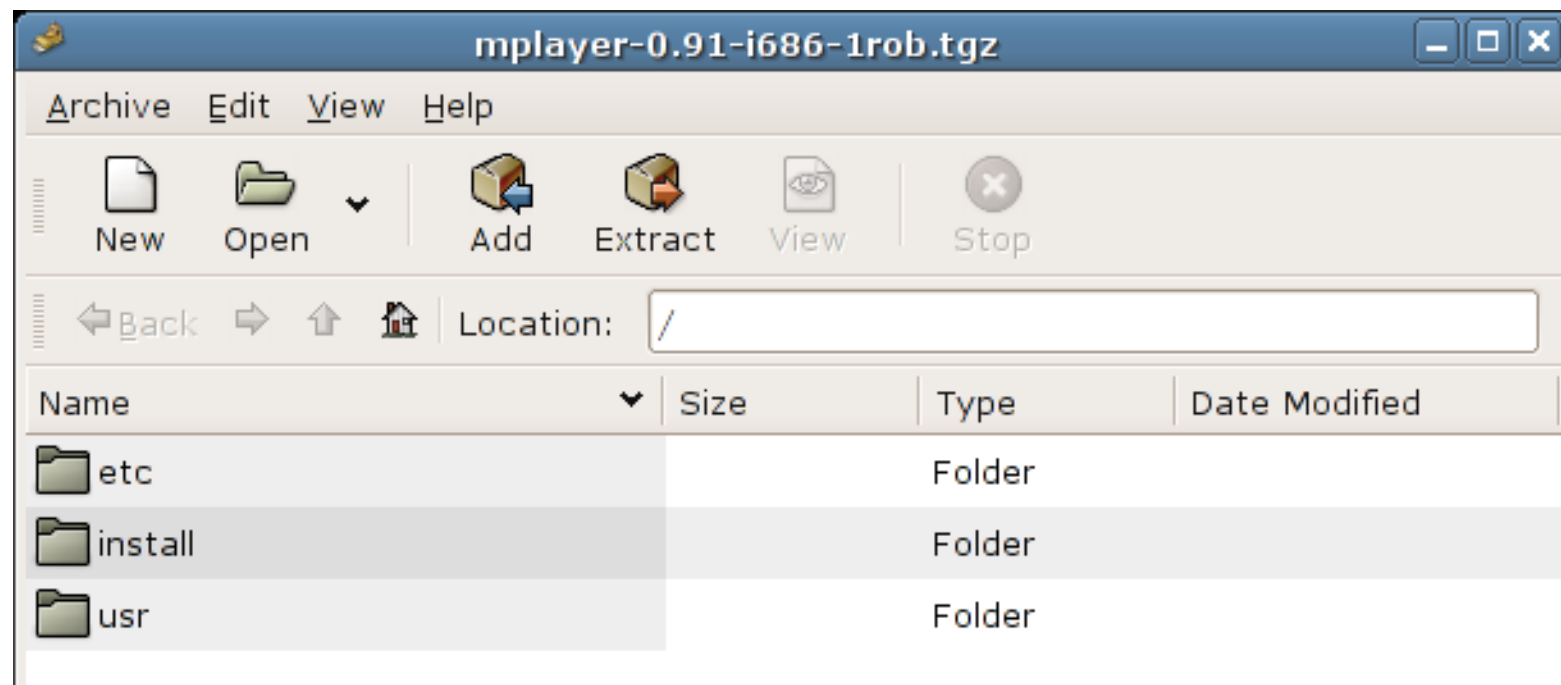
Ma:

- Le dipendenze?
- Come e stato compilato?

Pacchetti Slackware

Slackware gestisce il software mediante pacchetti con estensione .tgz.

Il software è precompilato ed inserito nel pacchetto con le directory a partire da / (root)



Pacchetti Slackware (2)

I tool ufficiali per la gestione dei pacchetti sono

- Pkgtools
- Installpkg
- Removepkg
- Upgradepkg

Tutti installano un pacchetto locale e non controllano le dipendenze.

Cosa sono le dipendenze?

- Altri pacchetti che forniscono servizi richiesti dal programma che vogliamo installare
- In Windows non ci sono dipendenze (a meno che non servano codec esterni)
- Neanche nel mac os

Dipende da come è stato compilato il programma..

Dipendenze (2)

È necessario

- Conoscere gli altri pacchetti (o i file) necessari
- Installarli prima del pacchetto corrente

Il meccanismo di gestione è sempre lo stesso, cambia la struttura del pacchetto

RPM e dipendenze

Gli rpm contengono una lista di nomi di file necessari per il funzionamento

Sarà poi compito del gestore di pacchetti (yum, Yast ecc) risalire dal nome del file mancante al nome del pacchetto.

Nome file → pacchetto

Pacchetti deb e dipendenze

I deb sono più complessi, tutte le informazioni sono contenute nel file.

- Una parte del pacchetto è deputata al controllo dei dati (control.tar.gz)
 - Configurazioni successive
 - Operazioni preinstallazione
 - altro...
- L'altra contiene i dati da installare (data.tar.gz)

Arch e dipendenze

Un pacchetto arch contiene soltanto i file da installare

Le dipendenze si trovano in un file sul repository (es `current.db.tar.gz`) insieme ad una descrizione del pacchetto

→ pacman installa i pacchetti contenuti nel file associato al pacchetto

Dove siamo?

Finora abbiamo visto:

- Come viene rilasciato un software
- Come alcuni pacchetti contengono informazioni sulle dipendenze

Cosa vedremo:

- Come i gestori di pacchetti operano
- Come e possibile ottimizzare il software compilandolo

Pacchetti arch

- Vengono gestiti da “pacman”
- `current / extra / testing / unstable`
`“/ community / release”`
- Se richiesto (flag `-Sy`) aggiorna la lista dei pacchetti e i file delle dipendenze (`current.db.tar.gz`, `testing.db.tar.gz` ...etc..etc..) dai repository specificati nel file `/etc/pacman.conf`
- Controlla le dipendenze e scarica i pacchetti dai repository

Pacchetti arch

- Salva i vecchi file di configurazione come: “nomefile.pacsave” quando si rimuovono o aggiornano dei pacchetti.
- Copia i file contenuti nel pacchetto nel file system
- Aggiorna il database locale dei pacchetti installati
- Gestisce le dipendenze inverse, ma solo se queste sono state installate insieme al programma che si sta per rimuovere.

ABS: Creazione di Pacchetti per arch

- Struttura del pkgbuild molto semplice
- Il pacchetto si crea lanciando un solo comando (“makepkg”)
- Possibilità di ottimizzare il pacchetto per il proprio sistema

```
# Maintainer: Super Bukki <bukki@bukki.org>
pkgname=bukki
pkgver=0.7
pkgrel=1
pkgdesc="Questo è un prog. utilissimo!"
url="http://www.bukki.org"
license="GPL"
depends=('xxx' 'yyy' 'zzz')
makedepends=()
conflicts=()
replaces=()
backup=()
install=
source=(http:// ... $pkgname-$pkgver.tar.gz)
md5sums=('8cd7 .... fcf1')
build() {
    cd $startdir/src/$pkgname-$pkgver
    ./configure --prefix=/usr
    make || return 1
    make DESTDIR=$startdir/pkg install }
```

Apt-get (Debian)

- Recupera i pacchetti dai repository contenuti in `/etc/apt/sources.list`
- Old stable / stable / testing / unstable / experimental
- Mantiene i file di configurazione quando si rimuovono pacchetti
- Leggendo la parte di controllo dal file esegue operazioni post installazione per la configurazione del software
- Non gestisce le dipendenze inverse

Aptitude (2)

- Già ora è consigliato l'uso di aptitude al posto di apt-get
 - Interfaccia grafica (ncurses)
 - Ingloba funzioni di ricerca e installazione
 - Gestisce meglio le dipendenze
- apt-build

Ebuild (gentoo)

- Un ebuild è un file di testo che contiene una descrizione del software, le istruzioni su come ottenerlo, configurarlo, compilarlo ed installarlo.
- Un ebuild in particolare contiene:
 - Informazioni sul creatore e sulla licenza
 - La lista dei pacchetti da cui dipende
 - Le flag USE
 - Info sulla versione (stable / masked / hard masked)
 - Dove scaricare i sorgenti

Ebuild (gentoo)

- Inoltre contiene gli script scritti in bash che vengono lanciati da emerge per la:
 - Decompressione del sorgente scaricato
 - Compilazione del sorgente
 - Installazione del programma compilato
 - Operazioni di configurazione post-installazione

Emerge (gentoo)

- EmERGE interpreta gli ebuild ed esegue le operazioni per installare il software nel sistema.
- Mantiene i file di configurazione quando si rimuovono pacchetti
- Non gestisce le dipendenze inverse...
 - **Ma Attenzione:** Quando si rimuove un pacchetto emerge NON controllerà se il pacchetto è richiesto da un altro pacchetto. Verrà solo emesso un avviso del fatto che la rimozione di pacchetti importanti potrebbe danneggiare il sistema.

Yast (SuSE)

- È il gestore dei pacchetti (ma non solo) di SuSE
- Basato su RPM
- Recupera i pacchetti da repository (media locali o ftp)
- Molto intuitivo
- Fornisce un servizio di aggiornamento windows-like per aggiornamenti di sicurezza o software “particolare”
- CHIEDI A FRAPPA

Costruzione pacchetti

Perchè crearsi i pacchetti?

- Non esiste ancora
- Voglio un pacchetto personalizzato
- Arch ad esempio si basa anche sui pacchetti fatta dalla community, dedicando loro un apposito mirror
- Evitare la ricompilazione (grossi pacchetti in gentoo)

Creazione di RPM

- Creare il file `/etc/rpmrc`

```
require_vendor: 1
    distribution: Red Hat
    require_distribution: 1
    topdir: /usr/src/packs
    vendor: NerSoft
    packager: nome creatore
    optflags: i386 -02 -m486
    optflags: alpha -02
    optflags: sparc -02
    signature: pgp
    pgp_name:
    pgp_path: /home/packages/.pgp
    tmppath: /tmp
```

Creazione di RPM (2)

- Creare un file .SPEC

```
Summary: Pacchetto di elaborazione matematica matrix
Name: matrix
Version: 1.0
Release: 1
Copyright: GPL
Group: Utility
Source: ftp://ftp.techno.it/matrix.tar.gz
%description
Progetto di Laboratorio. Parser per l'elaborazioni
  di matrici.
%files
%config /etc/matrix.cfg
/usr/bin/matrix
/usr/man/man4/matrix.4
```

Creazione pacchetti RPM (3)

- Copiare i file al loro posto
- `rpm -bb matrix-1.0.SPEC`
- `rpm -qi matrix`

Più compilato il pacchetto con i sorgenti

Creazione di deb

- Creare il file control

```
Package: acme
Version: 1.0
Section: web
Priority: optional
Architecture: all
Essential: no
Depends: libwww-perl, acme-base (>= 1.2)
Pre-Depends: perl
Recommends: mozilla | netscape
Suggests: docbook
Installed-Size: 1024
Maintainer: Joe Brockmeier [jzb@dissociatedpress.net]
Conflicts: wile-e-coyote
Replaces: sam-sheepdog
Provides: acme
Description:
```

Creazione di deb (2)

Sarà necessario creare script per

- Configurare il software dopo l'installazione (postinst)
- Consentirne la rimozione (prerm)

dopo aver creato l'albero di directory con tutti i file basterà eseguire

```
dpkg -b directory nomepacchetto.deb
```

Ottimizzazione del software

I sorgenti possono essere compilati con varie ottimizzazioni

- Precompilazione (`./configure --help`)
- Flag al compilatore (`gcc...`)
- Flag USE in gentoo

./configure

Si possono passare parametri per

- Abilitare / disabilitare feature
- Scegliere la directory di installazione
- Specificare il percorso per raggiungere eventuali dipendenze
- Controllare che i file necessari siano presenti nel sistema

Tutte le ottimizzazioni precompilazione passano di qua

Alcune Flag del compilatore

Il compilatore si occupa di tradurre il linguaggio di alto livello (C, C++, ...) in linguaggio macchina

gcc è in grado di ottimizzare il processo mediante alcuni flag

- -O num specifica il livello di ottimizzazione per l'architettura utilizzata
- -march="..." ottimizza per la marca del processore
- -pipe usa pipe per le comunicazioni tra i thread di compilazione
- Mille altre...

USE

- In un software ci sono alcuni moduli che possono non essere compilati ed inclusi negli eseguibili
- Tramite le USE si possono abilitare e disabilitare queste parti di programma
- Es `USE="X gtk gnome -qt -kde -java"`
- Per farlo configura opportunamente i parametri da passare allo script `configure`

Gentoo

- Gentoo punta tutto sull'ottimizzazione del sw, tutto deve essere compilato dopo aver impostato i parametri desiderati per il compilatore e le USE
- Emerge gestisce installazione, rimozione e aggiornamento del software mediante file ebuild
- Tutto deve essere compilato (tranne i pacchetti chiusi)