Sistemi Operativi

Unità 8: Altri Argomenti

Gestione dei Pacchetti Software

Martino Trevisan
Università di Trieste
Dipartimento di Ingegneria e Architettura

Argomenti

1. Perchè sono necessari
2. Package Manager
3. Pacchetti `deb` e Package Manager `apt`
4. Package Manager `snap`

Perchè sono necessari

Perchè sono necessari

Utenti e programmi

Appena installato, un SO contiene solo i programmi di default

- Utility per gestione del SO: `ls`, `ps`, `free`

Un utente vuole far girare le applicazioni che preferisce. Ha due opzioni:

1. Scrive un programma, lo compila e lo esegue
2. Usa un programma scritto da qualcun altro

L'opzione 2 è di gran lunga la più usata

Perchè sono necessari

Installazione

Per usare un programma, ci sono diverse opzioni:

- Scaricare il file binario del programma ed eseguirlo
  - Oppure un installer
  - In Windows: Scarico `installer.exe` e installo
- Scaricare e compilare il codice sorgente
- Usare un **Package Manager**
  - Come *AppStore* su iOS o Google Play di *Android*

Package Manager

Package Manager

Definizione

Un **Package Manager** è un software che si occupa di organizzare i software in uso in un sistema.
Ha l'obbiettivo di:

- Permettere l'installazione/rimozione di pacchetti software
- Verificare che il software non sia corrotto e arrivi da fonti sicure
- Gestire eventuali conflitti e dipendenze tra pacchetti
- Controllare gli aggiornamenti dei software installati

Un Package Manager scarica i pacchetti da un **Repository** pubblico

Package Manager

Tipologie

Monolitici: l'applicazione e le tutte dipendenze sono nello stesso pacchetto

- Come MacOS o Docker
- **Vantaggi:** ogni pacchetto si porta dietro tutto ciò che gli serve

A Pacchetti specifici: ogni pacchetto contiene un singolo software/libreria.
Quando si scarica l'applicazione, il Package Manager controlla e scarica eventuali dipendenze.

- Usato tipicamente in Linux: `apt`, `yum`
- **Vantaggi:** installazione veloce, no spreco di spazio su disco
- **Svantaggi:** gestire le dipendenze aumenta di **molto** la complessità!

Source-Based: il Package Manager scarica e compila il codice sorgente di ogni pacchetto

- Come `brew` usato su MacOS
- **Vantaggi:** programmi portabili su diverse architetture

Package Manager

Pre-installati nei SO

Fatti per l'utilizzo da parte di utenti non esperti

- Windows ⇒ Microsoft Store (precedentemente Windows Store)
- MacOS, iOS ⇒ AppStore
- Android ⇒ Google Play (precedentemente Android Market)

Caratteristiche:

- Closed-source
- Commerciali: offrono applicazioni a pagamento

Package Manager

Per la programmazione

Ambienti specifici hanno un **Package Manager** dedicato

- Python ⇒ `pip`, `conda`
- Java ⇒ `maven`
- JavaScript⇒ `npm`
- Go ⇒ `go get`

Package Manager

In Linux

Esistono due formati di **Package Binari**, ovvero che contegono software compilato:

- Pacchetti Deb: usati in Debian, Ubuntu
- Pacchetti RPM: usati in Red Hat, CentOS

I **Package Manager** installano **Package Binari** da repository pubblici:

- In Debian, Ubuntu: `apt`
- In Red Hat, CentOS: `yum`, ora rimpiazzato da `dnf`

Vengono usati tipicamente da riga di comando

Package Manager

In MacOS (oltre all'AppStore)

I software sono tipicamente in immagini **DMG**

- Formato per immagini di disco
- Contengono tutte le dipendenze

Si possono installare **Package Manager** aggiuntivi:

- `port` o MacPort
- `brew`

Entrambi scaricano i sorgenti e li compilano

Package Manager

Ogni **Package Manager** ha comandi diversi.

- Tipicamente si usano da riga di comando.
- Ma esistono interfacce grafiche per semplificare l'uso

Azioni comuni:

- `install`
- `remove`
- `update`
- `view dependencies`

Pacchetti `deb` e Package Manager `apt`

Pacchetti `deb` e Package Manager `apt`

Nei sistemi basati su Debian e Ubuntu si usa il formato **Deb**

- **Package atomici:** ognuno contiene un singolo software
- **Binari compilati:** si scaricano programmi già compilati **per la propria architettura**

Un pacchetto **Deb** è un archivio compresso contenente:

- I file binari
- Metadati: nome, versione
- Lista delle dipendenze
- Opzionalmente:
  - File di configurazione
  - Script da eseguire per installazione
  - Firma digitale GPG

Pacchetti `deb` e Package Manager `apt`

Utilizzo di `dpkg`

Il comando `dpkg` permette di gestire pacchetti **Deb**

- Installazione: `dpkg -i <file.deb>`
- Informazioni su un pacchetto: `dpkg -I <file.deb>`
- Disinstallazione: `dpkg -r <nome-pacchetto>`
- Lista di pacchetti installati: `dpkg -l`
- Lista dei file installati da un pacchetto installato: `dpkg -L <nome-pacchetto>`

Pacchetti `deb` e Package Manager `apt`

Da `dpkg` a `apt`

`dpkg` è un tool di basso livello

- Installa pacchetti da file `deb`
- Non risolve le dipendenze
- Non pratico da usare

Solitamente non si usa `dpkg` direttamente, ma ***Advanced package tool*** (`apt`)

- Risolve i problemi di cui sopra

Pacchetti `deb` e Package Manager `apt`

`apt`: repository

`apt` scarica package da repository online:

- Lista ottenuta dal file: `/etc/apt/sources.list` e da tutti i file nella cartella `/etc/apt/sources.list.d/`
- Repository pre-definiti quando si installa il SO
- Si possono aggiungere repository per package non presenti di default:
  - E.g., Chrome, Dropbox
- Un repository è identificato da un URL e ha dei tag
  Esempio:
  `deb http://it.archive.ubuntu.com/ubuntu/ focal main restricted`

Pacchetti `deb` e Package Manager `apt`

`apt`: comandi

Per installare paccheti con `apt` si usa il comando `apt` o `apt-get` (più vecchio ma analogo)

- Installazione: `apt install <nome-pacchetto>`
- Disinstallazione: `apt remove <nome-pacchetto>`
- Aggiornamento delle lista di pacchetti disponibili: `apt update`
- Ricerca di pacchetti nei repository: `apt-cache search`

Pacchetti `deb` e Package Manager `apt`

`apt`: dipendenze

Ogni volta che si installa un pacchetto, `apt` **risolve le dipendenze**

- Installa in automatico le librerie i software da cui dipende
- Problema complesso: generato un **grafo delle dipendenze**

Possono nascere **conflitti**, per problemi di versione

```
The following packages have unmet dependencies:
    package1 : Depends: package2 (>= 1.8) but 1.7.5-1ubuntu1 is to be installed
```

Tipicamente i pacchetti nei repository di sistema non hanno questi problemi

Pacchetti `deb` e Package Manager `apt`

`apt`: risoluzione dei problemi

In caso di dipendenze non risolte o altri problemi, si può dire ad `apt` di fare pulizia

- `apt autoclean`: elimina i pacchetti `.deb` scaricati relativi a versioni vecchie
  Nota: rimuove l'archivio Deb, che è inutile dopo installazione, ma viene tenuto in **cache**
  Non rimuove l'installazione
- `apt clean`: elimina tutti i pacchetti `.deb` in **cache**
- `apt autoremove`: **disistalla** elimina i pacchetti orfani, ovvero dipendenze installate per l'installazione di un'applicazione che poi rimuovete, così non sono più necessarie

Package Manager `snap`

Package Manager `snap`

`apt` e `snap`

`apt` funziona molto bene ed è usato con successo nella maggior parte dei sistemi Linux

- Economizza lo spazio: i pacchetti hanno dipendenze
- Le dipendenze sono installate e condivise da tutto il sistema

Nei sistemi Ubuntu, ora a fianco di `apt` si usa anche **Snap**

- Installato di default su Ubuntu, installabile anche su altre distribuzioni

Package Manager `snap`

Caratteristiche

Snap installa pacchetti **self-contained**

- Contengono il programma e tutte le dipendenze: librerie, altro software
- Di fatto contengono un File System in formato SquashFS
- Le applicazioni girano in una SandBox, con limitato accesso al sistema
- Concettualmente simile a un **container**!
  - Simile a Docker, ma pensato anche per utenti non esperti

Package Manager `snap`

Pro e contro

Vantaggi:

- Risolve problemi di dipendenze
- Maggiore sicurezza grazie a SandBox

Svantaggi:

- Si usa più spazio su disco
- Pacchetti sono più grandi da scaricare dalla rete
- Più pesante per il sistema:
  - Il File System di un pacchetto viene *montato* ad ogni avvio

Domande

A cosa serve un Package Manager?

- `A instradare i pacchetti di rete`
- `A installare i pacchetti software da repository pubblici`
- `A installare i programmi creati dall'utente`

Un pacchetto Deb contiene le tutte sue dipendenze:

- `Si`
- `No`

Un pacchetto Deb contiene i file sorgenti:

- `Si`
- `No`

Il Package Manager `apt` installa le dipendenze:

- `Automaticamente`
- `Mai`
- `Su richeista`

Un pacchetto Snap contiene le tutte sue dipendenze:

- `Si`
- `No`