Distribuzione del software

Enrico Tassi

(slides originali di Stefano Zacchiroli)

Dipartimento di Scienze dell'informazione, Università di Bologna

March 9, 2009

Applicativi distribuiti in forma sorgente (1/2)

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:

- donwload dell'archivio dei sorgenti (client http, ftp, . . .)
- estrazione dei sorgenti (tar, gzip, bzip2, . . .)
- configurazione dei sorgenti (autoconf, configure, . . .)
  - verifica automatica delle dipendenze (./configure)
  - risoluzione manuale dei problemi derivanti (sysadm)
- compilazione da sorgenti a binari (make)
- installazione dei binari (make)

Applicativi distribuiti in forma sorgente (2/2)

Vantaggi

- incoraggia portabilità e studio dei sorgenti
- flessibilità nella scelta delle opzioni di configurazione e compilazione (e.s. ottimizzazioni di gcc per l'architettura sottostante)
- paranoia / control freak

Svantaggi

- non uniformità della procedura di installazione (autotools, make, scons, Cmake, . . . simili ma diversi)
- lunghi tempi di compilazione
- necessità di applicativi e librerie a compile-time, non necessari a run-time (es i .h)
- 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 via sorgente)

- installazione facile e veloce
- librerie linkate staticamente o distribuite con il binario
- upgrade “drop in”
- niente tempo di compilazione

Svantaggi

- dipendenza dall’architettura
- dipendenza da librerie dinamiche
- fiducia in chi ci fornisce il binario

Pacchetti

Per superare gli svantaggi della distribuzione degli applicativi, ogni distribuzione GNU/Linux offre l'astrazione del "pacchetto"

- i pacchetti definiscono la granularità alla quale è possibile aggiungere e rimuovere componenti del sistema
- 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
- 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: sorgente e binario

- 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 sorgente 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:

- rilascio della versione X di una componente
- creazione della versione X del pacchetto sorgente
- compilazione delle versioni X dei pacchetti binari (es. un pacchetto binario per ogni architettura supportata)

Utilizzo di un pacchetto

Il bello di tutto ciò è che l'amministratore di sistema non deve necessariamente interagire con la vita di un pacchetto, e può dedicarsi ad altro:

- 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

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 (o può essere) compilato una sola volta (per architettura)

Svantaggi

- fiducia nei manutentori della distribuzione
- rigidità nella scelta delle opzioni di configurazione

Riferimenti: appunti, capp.38-39

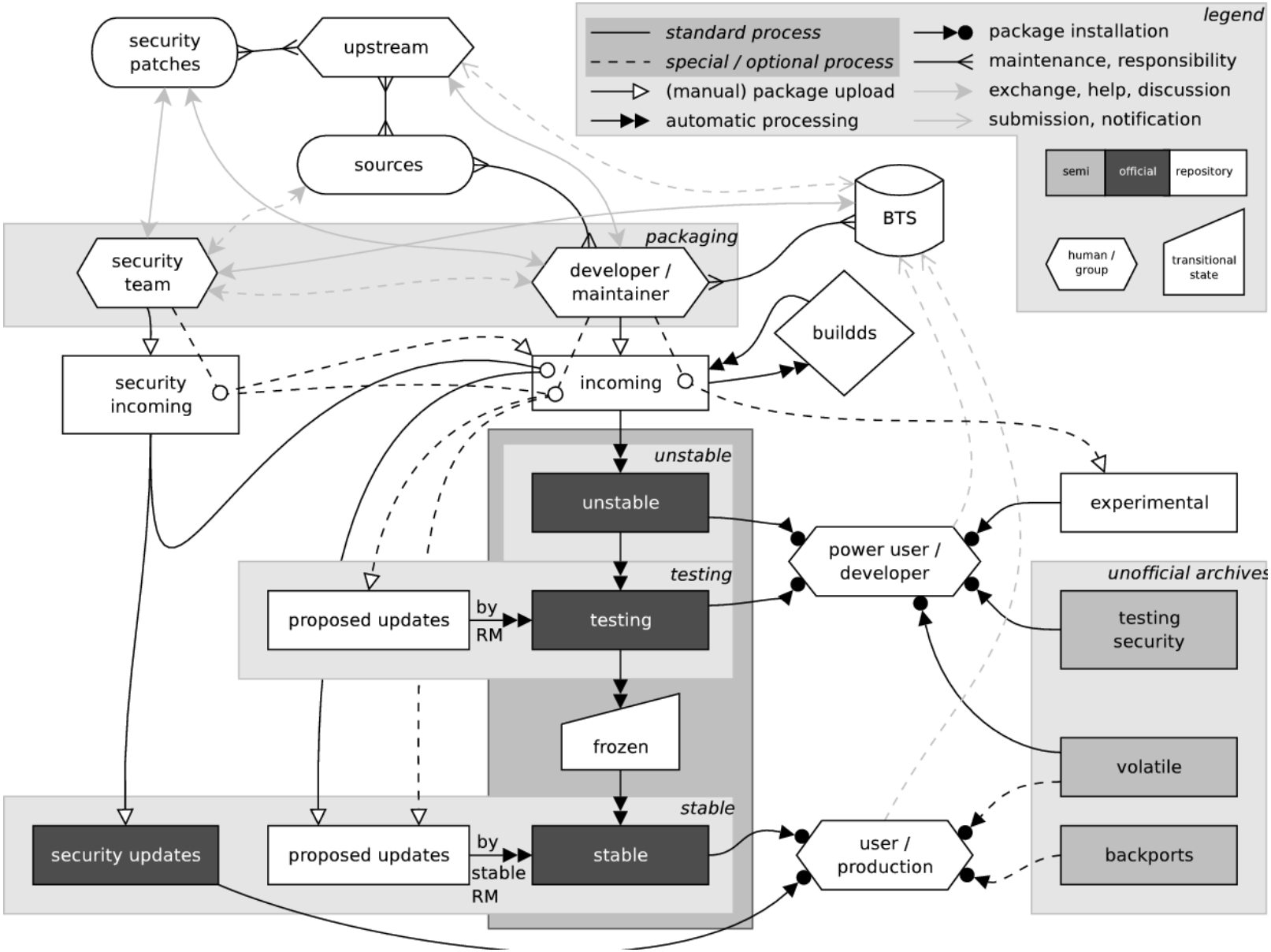
Case study: Debian, panoramica

Panoramica sui pacchetti `.deb`

- 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, reconfigure
- Gestione dei pacchetti, la gerarchia Debian:
  - dpkg-deb $\Leftarrow$ dpkg $\Leftarrow$ apt-get $\Leftarrow$ $\{$ dselect ; aptitude ; synaptic ; $\ldots$ $\}$

Case study: Debian, distros e workflow

Debian workflow, Copyright Martin F. Krafft



Case study: Debian, repository

Le repository di pacchetti `.deb`

- In `/etc/apt/sources.list` o `/etc/apt/sources.list.d`
- Formato `TIPO URL SECTIONS`
- Tipo: `deb` `deb-src`
- Url: `http://ftp.it.debian.org` oppure `file:///my/local/path`
- Sections: `main contrib non-free` oppure `./`

Case study: Debian, repository — esempio

```
deb http://ftp.it.debian.org/debian/ sid main contrib non-f
deb http://ftp.it.debian.org/debian/ testing main contrib n

deb-src http://ftp.it.debian.org/debian/ sid main contrib

deb http://www.debian-multimedia.org sid main

deb http://security.debian.org/ testing/updates main

deb file:///srv/local-debian/ ./
```

Case study: Debian, comandi (1/2)

I comandi principali per manipolare i pacchetti

- dpkg
  - Basso livello, installazione e rimozione del .deb, non risolve le dipendenze, non scarica i pacchetti dai repository
  - `dpkg -i foo_1.0-1.deb`
  - `dpkg -r foo`
- apt-get
  - Medio livello, risolve dipendenze, conosce le repository
  - `apt-get update`
  - `apt-get install foo`
  - `apt-get install foo=version`
  - `apt-get remove foo`
  - `apt-get autoremove`
- deborphan
  - Alto livello, interfaccia interattiva, collezione pacchetti non utilizzati (soprattutto librerie)
  - `orphaner`

Case study: Debian, comandi (2/2)

I comandi principali per manipolare i pacchetti

- aptitude
  - Interfaccia testuale o grafica (non ancora ufficiale)
  - Programma interattivo, alto livello, tab, ricerca, filtraggio
- apt-cache
  - Ricerca nei metadati dei pacchetti
  - `apt-cache search pdf editing`
  - `apt-cache show pdfedit`
- ept-cache / debtags
  - Ricerca "smart" nei metadati dei pacchetti (usando tag)
  - `http://debtags.alioth.debian.org/ssearch.html`
  - `ept-cache related gnome`
  - `debtags search x11::xserver`
- apt-file
  - Ricerca del pacchetto che contiene un file specificato
  - `apt-file update`
  - `apt-file search dos2unix`

Case study: Debian, manuali

Riferimenti

- appunti, cap. 42—46
- Debian documentation `http://www.debian.org/doc/`
- "The Debian System, Concepts and Techniques", Martin F. Krafft, Open Source Press, `http://debiansystem.info/about/book`

Esercizi

- Installa il pacchetto `sl`
- Controlla il contenuto del pacchetto `sl`
- Chiedere al sistema che pacchetto contiene il file `a2ps.1.gz`
- Aggiungere la repository di `http://debian-multimedia.org` per la distribuzione e architettura giusta
- Aggiornare la lista dei pacchetti
- Verificare che vi siano più pacchetti disponibili (es. acroread)
- Rimuovere la repository appena aggiunta (è tutta roba non-free :(
- Installare il pacchetto `vrms` ed eseguire l'utility che fornisce per essere sicuri di non aver installato niente di non-free
- Leggere le istruzioni di backports.org
- Leggere le istruzioni di `http://volatile.debian.org`