AMMINISTRAZIONE DI SISTEMI

Pierluca Pevere

FUNZIONAMENTO DEL SISTEMA

GESTIONE DEI PACCHETTI SOFTWARE

Ogni pacchetto software ha un ciclo di vita composto da:

1. installazione
2. aggiornamento
3. disinstallazione

Si possono però riscontrare porblematiche, ad esempio: prerequisiti hardware o di sistema operativo, dipendenze da o di altri componenti software oppure la configurazione.

INSTALLAZIONE MANUALE

L'installazione manuale di pacchetti software può avvenire:

- da binari:
  - semplice copia nei posti giusti
  - verificare manuale della compatibilità con l'architettura
  - verifica manuale delle dipendenze
- da sorgente:
  - necessità di compilazione
  - indipendenza dall'architettura
  - possibile maggior flessibilità nel soddisfacimento delle dipendenze

  più frequente e migliore rispetto alla precedente perchè si compila su una determinata architettura e non si è vincolati a quella presente sul computer che ha compilato il binario

Un aspetto importante sono le **dipendenze** del componente software da altri.
Nel caso di un'installazione da binari, probabilmente è necessario disporre dei software indicati come prerequisiti in una **versione specifica.
Nel caso di installazione da sorgente qualche grado di flessibilità in più lo si ha. Infatti vi è la possibilità che i sorgenti dispongano di diverse interfacce per adeguarsi a cosa si trova sul sistema. C'è però la necessità di disporre non solo dei componenti runtime relativi ai software richiesti, ma anche delle librerie di sviluppo: in un sistema ideale ho tutti i sorgenti per cui dispongo sempre di tutti questi elementi. Nelle distribuzioni, per flessibilità, ogni pacchetto software ha un corrispondente paccetto `-dev` o `-devel`

Installazione manuale tipica in Linux

Il caso più comune è quello di software distribuito per mezzo di un archivio `.tar.gz`, scritto in C, predisposto alla compilazione tramite il framework `autoconf`.
`autoconf` verifica se sono soddisfatti tutti i prerequisiti, rileva le versioni e le collocazioni dei pacchetti sul sistema, accetta dall'utente la specifica di varianti (attivazione/disattivazione di funzionalità, preferenze architetturali, ...), genera i `Makefile` sulla base delle specificità del sistema e delle scelte operate dall'utente.

I passi tipici sono:

1. reperimento del software
2. estrazione del pacchetto
3. esame delle scelte disponibili
4. configurazione dei sorgenti
5. compilazione
6. installazione

N.B.: solo quest'ultima operazione può richiedere i diritti di superutente, e quindi si deve evitare di compiere le precedenti come root. Sono noti casi di malware che sfruttano proprio la cattiva abitudine di eseguire una o più delle operazioni preliminari con diritti eccessivi.

Autenticazione La PRIMA cautela da usare quando si scarica software dovrebbe essere quella di verificarne l'autenticità da una firma digitale, ovviamente server una chiave pubblica fidata per compiere questa operazione.
Un esempio può essere:

```
# mostra il key id
gpg --verify FILE.asc FILE.tar.gz
# l'autenticità della chiave in questo caso deriva solo dalla fonte (a volte non basta)
gpg --keyserver pgpkeys.mit.edu --recv-key <KEY_ID>
# si deve valutare caso per caso, successivamente si ripete il primo comando
```

Come minimo dovrebbe essere disponibile un fingerprint.
Basta mettere il fingerprint (ad esempio in formato `.sha256`) nella stessa directory del file da verificare e lanciare

```
sha256 -c FILE.sha256
```

Per quanto riguarda l'**estrazione del pacchetto**, che solitamente si presenta sotto forma di archivio `tar` compresso, è buona prassi determinare una collocazione sensata per i sorgenti ed estrarre in tale directory l'archivio. Nel caso si stia per affrontare un upgrade sostanziale del sistema, che coinvolge numerose applicazioni, può essere utile raccogliere in modo più chiaro tutti i pacchetti che verranno installati unitariamente. È prudente, infine, testare l'archivio prima dell'estrazione per verificare la gerarchia di directory generata (questo per evitare che il creatore del pacchetto abbia messo direttamente i file e non una cartella che li contiene e quindi estraendo si andrebbe ad intasare la cartella senza poi sapere quali fossero i file utili per quel pacchetto). Ad esempio:

```
cd /usr/local/src
# per testare il pacchetto
tar tvzf net-snmp-5.4.tar.gz
# per estrarre tutto
tar xvzf net-snmp-5.4.tar.gz
```

La fase successiva consiste nell'**esame delle scelte disponibili**: si entra nella directory generata dall'estrazione e si esamina il contenuto (è bene leggere i file `README` ed `INSTALL` che solitamente sono forniti col software). Successivamente, se esiste, si lancia l'eseguibile di nome `configure` con il parametro `--help` per ottenere una lista dei parametri di configurazione disponibili: scelte comuni riguardano la collocazione del software, l'attivazione o la disattivazione di sottocomponenti, la predisposizione dei componenti attivati come moduli dinamicamente caricabili piuttosto che la loro integrazione statica nel codice, ...

Si procede, quindi, con la **configurazione dei sorgenti** rilanciando `configure` con i parametri scelti. Potrebbero sorgere problemi evidenziati dall'esecuzione del file di configurazione (tipicamente l'assenza di pacchetti necessari come prerequisiti), tuttavia `configure` non è a prova di errore e quindi può servire un'indagine manuale che, talvolta, risulta complessa.
Per limitare l'insorgere dei problemi è utile, se non necessario, seguire le indicazioni presenti nei `README` ed `INSTALL`.
In ogni caso la procedura genera un `config.log`.

Si è arrivati, a questo punto, alla **compilazione** tramite `make` o le indicazioni presenti nell'output del passo precedente.

Ed, infine, con l'**installazione** tramite `sudo make install`.

Questo tipo di installazione offre la possibilità teorica di verificare il codice, se non lo si fa è un rischio per la propria sicurezza, difatti, se ci si fida della firma sull'archivio non è diverso rispetto a verificare la firma su un binario. Tuttavia, è molto più difficile da manutenere e richiede molti componenti ausiliari (come headers, librerie, processori di macro, compilatori, linker ...) che possono avantaggiare un attaccante.
Per questa serie di motivi entrano in scena le distribuzioni e i pacchetti:

- chiavi di verifica installate una volta per tutte
- gestione automatica delle dipendenze
- garanzia di compatibilità binaria tra tutti gli elementi del set

INSTALLAZIONE ASSISTITA

Comunemente effettuata per mezzo di software ausiliari:

- package manager specifico della distribuzione Linux (rpm/yum, dpkg/apt, ...)
- installer per Windows

Un tool di installazione (come quelli elencati sopra) può farsi carico delle verifiche relative alle dipendenze ma non può configurare ogni dettaglio del sistema in modo specifico, tuttavia può generare dinamicamente dati specifici.

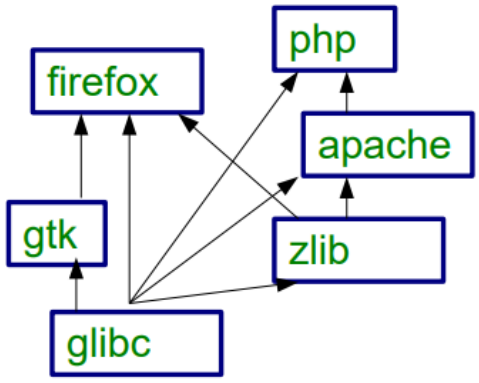

Figure 1: A $\rightarrow$ B significa che A "serve" per B; "serve" può essere una dipendenza tra funzionalità logiche (non ha senso avere un linguaggio di generazione pagine web senza un web server) o fisiche (un binario linkato dinamicamente non gira senza tutte le librerie di cui importa i simboli)

PACCHETTI

Le distribuzioni Linux organizzano il software in pacchetti che dispongono di un package manager per la loro gestione.
Un pacchetto si presenta sotto forma di singolo file che contiene in forma compatta un insieme di:

- software precompilato
- criteri per la verifica della compatibilità e dei prerequisiti
- procedure pre/post installazione

La verifica della compatibilità con un determinato sistema può essere data solo a patto di vincolare con precisione i seguenti parametri:

1. architettura
2. versione della disto
3. versione del software contenuto nel pacchetto

DISTRIBUZIONI: criteri per la scelta

Architetture supportate

Tutte le distribuzioni supportano i processori Intel a 32 bit, la maggior parte anche quelli a 64 bit, alcune sono disponibili per tutte le varietà di processori su cui è stato portato il kernel. È bene ricordare che i pacchetti di terze parti potrebbero, tuttavia, non essere disponibili per le architetture supportate.

Stabilità vs. aggiornamento

Il processo di rilascio frequente e continuo del software nel mondo GNU/Linux ha come conseguenza inevitabile che le versioni più aggiornate possano essere meno stabili. Vi sono distribuzioni che hanno come filosofia l'inclusione dei pacchetti più recenti (e quindi con funzionalità maggiori e meno bug) anche a costo di una minor robustezza, ed altre che garantiscono l'inclusione solo di software ben collaudato.

Version vs rolling

Alcune distribuzioni sono "versionate": durante il ciclo di vita di una versione vengono forniti solo aggiornamenti correttivi, tutte le novità vengono testate e accumulate per la pubblicazione in una nuova versione (che andrebbe installata sovrascrivendo la versione precedente).
Altre sono "rolling": ogni volta che c'è una nuova funzionalità viene testata e distribuita, quindi in ogni momento il sistema è alla versione più recente.
Un esempio di distribuzione versionata è Ubuntu: viene rilasciata ogni 6 mesi (ad aprile ed ottobre) e il supporto per la versione precedente è interrotto al rilascio della nuova versione, tuttavia le versioni dispari degli anni pari che vengono etichettate come LTS (Long Term Support) che hanno un supporto per gli aggiornamenti di 5 anni.

Supporto e durata

La disponibilità di supporto garantito è tipica delle distribuzioni commerciali, ma anche con le distribuzioni gratuite più diffuse, in virtù della dimensione della relativa comunità di utenti, è semplice risolvere eventuali problemi.
Per installazioni di tipo server esistono varianti LTS: per 5/7 anni chi cura la distro garantisce che gli aggiornamenti non modifichino le API (tipicamente viene garantito solo il backporting dei security fix, non quello di tutti i bug fix).

Ampiezza del set di pacchetti
Si va dai 1500 delle distro minimali ai 26000 di Debian.
Una scelta intelligente mette tutto l'essenziale in 1 CD.

DEBIAN e REDHAT

Due distribuzioni capostipite da cui sono state derivate quasi tutte le varianti più diffuse di Linux sono proprio Debian e Red Hat.
Hanno 2 sistemi di gestione dei pacchetti con molte somiglianze:

- Tool di basso livello per la gestione dei singoli pacchetti
- Tool intermedi per la gestione coordinata di pacchetti e dipendenze
- Tool per il riempimento automatico da repository dei pacchetti necessari

Pacchetti
I pacchetti per le distro Debian e derivate (es. Ubuntu) sono in formato `.deb`: `aptitude-0.2.15.9-2_i386.deb`

I pacchetti per le distribuzioni RedHat e derivate (es. CentOS, Fedora) sono in formato `.rpm`: `httpd-2.4.6-45.el7.centos.x86_64.rpm`

I nomi hanno una semantica specifica:

- `aptitude` e `httpd` sono i nomi dei pacchetti
- `0.2.15.9` e `2.4.6` sono le versioni del software
- `2` e `-45.el7.centos` sono le versioni del pacchetto
- `i386` e `x86_64` sono le architetture