RPM per chi conosce InstallShield®

Non mettete "echo Avanti", che si vede che venite da Windows.

Università degli Studi di Modena e Reggio Emilia

Rivolto a sviluppatori che ...

- Hanno una competenza pregressa acquisita su piattaforma Windows
- Sono interessati ad acquisire competenza sulla piattaforma Linux
- Hanno un *corpus* di codice C/C++ che si può ragionevolmente pensare di portare
- Desiderano che l'installazione *non* sia un problema

Panoramica generale

- Ci sono differenze di *filosofia* tra le due piattaforme
- Ci sono differenze di *filosofia* nell'installazione del software
- Inutile nuotare controcorrente, in particolare quando si dispone di un motoscafo
- Occorre leggere le istruzioni

Filosofia Linux/Unix in pillole 1

- Molti strumenti specializzati
  - Invece di uno strumento unico
- Combinazione di strumenti specializzati
  - Invece di copia e incolla
- Le applicazioni sono meno importanti
  - Facile sostituire strumenti piccoli
  - Ricca dotazione di strumenti free

Filosofia Linux/Unix in pillole 2

- Gerarchia di directory definita dal sistema
  - Filesystem Hierarchy Standard, FHS
  - Tutti i programmi insieme
  - Tutte le configurazioni insieme
  - Non esiste *la* directory del programma
  - Non occorre registrare i componenti
- Configurazione in file di testo
  - Interfaccia grafica opzionale
  - Occorre leggerli oltre che scriverli

Filosofia Linux/Unix in pillole 3

- Versioni multiple distinte delle librerie
  - Tutte le librerie insieme
  - Il nome include la versione
    - Nozione di compatibilità delle interfacce
    - *Major* e *minor* version
  - Sviluppo e distribuzione
    - Sviluppo: `libbonobo.so` -> `libbonobo.so.2`
    - Trovo: `libbonobo.so.2` -> `libbonobo.so.2.0`
    - Poi: `libbonobo.so.2 -> libbonobo.so.2.1`

Perchè RPM

- Formato standard Linux Standards Base, LSB
  - Le differenze con altri formati sono modeste
- Piuttosto diffuso
  - Tool di conversione verso altri formati
- Alcuni limiti
  - Qualità spesso modesta dei singoli pacchetti
  - Aggiornamento automatico visto con sospetto

Filosofia RPM

- Il controllo non è dell'applicazione
  - Il sistema ha il controllo
  - Script e sicurezza
- Meglio non installare
  - Che installare qualcosa che non funziona
- Nozione di *pacchetto* RPM
  - Ovvero `ppp-2.4.1-3.i386.rpm`
  - Pacchetti multipli piuttosto che opzioni

Vantaggi per gli utenti di RPM

- Controllo delle dipendenze
  - Non installa se mancano i requisiti
- Aggiornamento e disinstallazione
  - Non restano indietro pezzi e refusi
  - Salvaguardate le configurazioni modificate
- Verifica e metadati
  - Segnala modifiche o installazioni corrotte
  - Risale dal singolo file al pacchetto

Vantaggi per chi sviluppa con RPM

- Controllo delle dipendenze
  - Associa a un pacchetto i requisiti perchè funzioni
- Aggiornamento e disinstallazione
  - Difficile lasciare refusi
  - Non è indispensabile migrare la configurazione ma non guasta
- Comodità
  - Integrato con il processo di sviluppo
  - Si prova altrove e poi si rimuove senza danni

Sorprese

- Non interattivo
  - Niente *splash screen*
  - Nessun dialogo (preferenze, licenza ...)
  - Nessun input
- Formato pacchetto fissato a priori
- Raramente occorrono script
  - Eccetto componenti di sistema
  - Eccetto chi nuota contro corrente

Come si fa un pacchetto

- Istruzioni nello *spec file*
  - Metadati: descrizione, versione ...
  - Dipendenze
  - Sorgenti
  - Istruzioni di compilazione
  - Lista dei file e dove si installano
  - Eventuali script
- Lavorare nella propria `$HOME`

L'ambiente di lavoro in $HOME

- Predisporre la directory di lavoro

```
$HOME/rpm
$HOME/rpm/BUILD
$HOME/rpm/RPMS/{i386,i486,i586,i686,noarch}
$HOME/rpm/SOURCES
$HOME/rpm/SPECS
$HOME/rpm/SRPMS
$HOME/rpm/tmp
```

- Configurare `.rpmmacros`

```
%_topdir  /home/utente/rpm
%_tmppath /home/utente/rpm/tmp
```

Descrizione, versione e macro

- In cima le definizioni

```
%define name il-nome-del-pacchetto
%define version 1.0.3
%define release 6
```

- Informazioni di servizio

```
Name: %{name}
...
Summary: una riga di descrizione
Group: qui copiate un gruppo appropriato
License: non lasciate in bianco
URL: non lasciate in bianco
```

Come si fa un pacchetto

- Indicare sorgenti e patch

```
Source %{name}-%{version}.tar.gz
Patch miemodifiche-%{name}-%{version}.patch
Patch1 altre-modifiche.patch
```

- Sorgenti e patch in `$HOME/rpm/SOURCES`
- Indicando una URL viene rimossa
- Indicate la *build root* di pseudo installazione

```
BuildRoot: %{_tmppath}/%{name}-root
```

Dipendenze: requisiti

- Automatiche (*shared object*)
  - `Require: glibc.so.6(GLIBC_2.0)`
  - La `glibc` è una fede, per il resto fidarsi è bene ...
- Manuali su pacchetti, magari con versione

  `Require: zlib >= 1.1.3`
- Minimizzare le dipendenze

  `BuildReq:` *`requisiti solo al build, p. es. header`*

  `PreReq:` *`prima di eseguire gli script`*
- Dipendenze sui singoli file

Dipendenze: soddisfarle

- Implicite: se stessi
- Automatiche (*soname*)
- Manuali: pacchetti *virtuali*

```
Provides: sendmail
```

- Trappole
  - Librerie degli sviluppatori troppo recenti
  - Dipendenze incrociate invocando se stessi negli script

Compilazione

- Pulizia

```
%prep
rm -rf $RPM_BUILD_ROOT
```

- A posto i sorgenti

```
%setup -q
%patch -p0
%patch1 -p1
```

- Configurare e compilare

```
%build
./configure --prefix=/opt/%{name} opzioni
make variabili
```

Percorsi e configurazione

- Bisogna configurare per la destinazione reale

```
--prefix=/opt/%{name}
```

- Installare i file nella *build root*

```
%install
mkdir -p $RPM_BUILD_ROOT/{bin,man}
make install DIR_INSTALLAZ_PROGXXX=$RPM_BUILD_ROOT/bin
install unprogramma $RPM_BUILD_ROOT/bin/unprogramma
```

- Pulizie finali

```
%clean
rm -rf $RPM_BUILD_ROOT
```

Elencare i files

- Indicare proprietari e permessi

```
%attr(0755, root, bin) /opt/%{name}/bin/unprogramma
%defattr(0644, root, bin)
```

- Indicare più file

```
%attr(0644, root, bin) /opt/%{name}/data/*.dat
%attr(0644, root, bin) /opt/%{name}/data
%dir /opt/%{name}/data
```

- Opzioni di verifica

```
%attr(0644, root, bin) %verify(not group) /opt/%{name}/cache
```

Come si fa un pacchetto

- Indicare la documentazione

```
%doc /opt/%{name}/LEGGIMI.TXT
```

- Indicare i file di configurazione *uno per uno*

```
%config /etc/opt/%{name}/main.cf
%config(noreplace) /etc/opt/%{name}/custom.cf
%config(missingok) /etc/opt/%{name}/current.cf
%ghost /var/opt/%{name}/logs
```

- Sfruttare una lista di file esterna

```
%files -f MANIFEST
```

Eventuali script

- Sezioni dello *spec file*

  `%pre`

  `%post`

  `%preun`

  `%postun`

  `%trigger`

- Problemi
  - Gli *shell script* sono un'arte
  - Non tutte le `sh` sono `sh`
  - Non conviene avviare demoni

Provare, provare, provare

- Usare `rpmlint`
  - Richiede un minimo di impegno
- Verificare il pacchetto sull'host stesso
  - Usando `rpm -Vp` ...
- Provare su un altro host
  - Con i prerequisiti deve riuscire
  - Senza deve fallire

Firma crittografica

- Basata su `gpg`
- Impostata in `.rpmmacros`

```
%_signature gpg
%_gpg_path ~/.gpg
%_gpg_name Nome chiave
```

- Rendere disponibile la chiave pubblica
  - Sul sito
  - Sul CD-ROM

Altre caratteristiche

- Pacchetti multipli da un solo sorgente
  - Client e server
  - Librerie e header
- Integrazione con ambienti desktop
  - GNOME e KDE
  - Comparire nei menu

Dopo i primi passi ...

- Verso un miglior processo di sviluppo
  - Sorgente, p. es. Itanium

    `autoconf`

    `automake`

    `libtool`
  - Infrastruttura di unit testing
  - Controllo di versione con CVS
  - Regressioni
  - Rebuild automatico e *farm*

Link

- Il sito RPM

  http://www.rpm.org

- Libro on line "Maximum RPM"
- Mandrake RPM HOWTO
- Articoli IBM DeveloperWorks

  http://www-106.ibm.com/developerworks/library/l-rpm1