Installare programmi su Unix e Linux

Usare pacchetti (rpm) per l'installazione di programmi su Linux

Per installare dei programmi su Linux esistono vari modi:
- compilare il sorgente, pratica che può essere complessa ma è utile in casi particolari;
- utilizzare pacchetti (packages) che contengono i programmi già compilati e pronti per l'uso, facilitando e standardizzando la gestione del software sul sistema.

I sistemi di package più comuni su Linux sono RPM e DEB.
I pacchetti **.deb** vengono usati nelle distribuzioni derivate da Debian, gli **.rpm** sono stati definiti da RedHat e risultano essere i più diffusi.
Slackware pacchettizza il suoi programmi con normali tar gzippati: **.tgz**.

Affrontiamo qui l'uso di RPM, Red Hat Package Manager, sottolineando che file .deb e, in parte, .tgz vengono gestiti con comandi diversi ma hanno una logica simile.

Un package costruito con RPM è un archivio di file e informazioni che può essere installato, rimosso, interrogato sul sistema.
RPM permette di installare programmi, già compilati, con una facilità e rapidità estrema sul proprio Linux (è paragonabile ad un unico **setup.exe** su Windows).
Si sottolinea che ogni distribuzione e anche ogni versione della stessa distribuzione richiede pacchetti dedicati, adatti per il proprio sistema: un RPM realizzato per RedHat 6.2, per esempio, difficilmente funzionerà su RedHat 7.2.
RPM gestisce automaticamente le "dependencies": se si prova ad installare un RPM che richiede librerie o programmi non presenti o non abbastanza aggiornati sul sistema, l'installazione fallisce e viene indicato quali file mancano.
Analogamente, se si prova a rimuovere un package che contiene file utilizzati da altri programmi, viene dato un messaggio di errore.
Gli RPM automaticamente distribuiscono i file di un pacchetto nelle directory giuste (logs in /var/log, file di configurazione in /etc/, binari in /usr/bin o /usr/sbin, script di startup in /etc/rc.d/init.d/ ecc.) e verificano la presenza di conflitti o installazioni più recenti.
La rimozione di un RPM non cancella mai nulla che non abbia installato. Se deve sostituire o cancellare un file di configurazione, per esempio, viene mantenuto il file esistente con il suffisso `.rpmsave`.

Le opzioni più comuni per usare il comando `rpm` per gestire file **.rpm** sono:
`rpm -i [opzioni] pacchetto` Installa il pacchetto .rpm specificato.
`rpm -U [opzioni] pacchetto` Aggiorna il pacchetto con una versione più recente.
`rpm -e [opzioni] pacchetto` Disinstalla il pacchetto, rimuovendone i file dal sistema.
`rpm -q [opzioni] [pacchetto]` Visualizza informazioni varie sul pacchetto (descrizione, file contenuti ecc.)

Le comuni distribuzioni Linux offrono svariati tool grafici per una semplice gestione dei pacchetti installati sul sistema. Di fatto questi programmi eseguono le stesse operazioni del comando rpm, ma sono più semplici ed immediate da usare.
La tendenza, sempre più diffusa, è quella di prevedere meccanismi di update automatizzato, per gestire il sempre alto numero di aggiornamenti (per sicurezza e bug fix) di programmi.
E' un principio analogo al **Windows Update** su sistemi Microsoft, ma si applica a tutti i programmi installati, e non solo al sistema operativo.

Nomenclatura di pacchetti .rpm

Trovare l'rpm giusto per la propria distribuzione Linux può non essere semplice, come non lo è sempre capire quale rpm contiene il programma o il file che ci servono.

Esiste una convenzione standard per dare un nome ad un package RPM:
`efax-0.8a-11.i386.rpm` corrisponde a `nome-versione-release.arch.rpm`
- **nome** è generalmente il nome del programma fornito col pacchetto,
- **versione** indica la versione del programma, secondo la convenzione usata da chi lo ha realizzato,
- **release** è la release del pacchetto, può anche indicare la distribuzione (es: **mdk** indica un RPM per Mandrake),
- **arch** è l'architettura hardware per la quale il pacchetto è stato previsto (**i386**: sistemi Intel 386 o superiori; **i686**: ottimizzato per sistemi Pentium 2 o superiori; **sparc**: Sun Sparc; **src**: sorgenti da compilare;),
- **rpm** è l'estensione predefinita.

Se il nome di un pacchetto contiene il suffisso **-devel**, nel rpm ci sono librerie utilizzate dal programma indicato o da altri pacchetti che dipendono da questo.
A volte per installare un programma compiutamente sono necessari più pacchetti .rpm, spesso se un programma ha natura modulare, c'è un pacchetto diverso per ogni modulo.

Gestire pacchetti RPM e DEB

Distribuzioni basate su RPM (RedHat, Mandriva, Suse...) e su DEB (Debian, Ubuntu...) utilizzano comandi diversi per fare gestire i pacchetti.
Vediamo in breve come si fanno le stesse operazioni nei due mondi.

- **Software e comandi**

dpkg e **rpm** sono i comandi base per gestire (installare, rimuovere, interrogare) pacchetti di tipo .deb e .rpm (rispettivamente).
apt-get, **yum** (ma anche urpmi su Mandrake, o up2date su RedHat o Yast2 su Suse ecc.) sono sistemi per gestire automaticamente lo scaricamento di pacchetti da Internet, il loro aggiornamento, la gestione delle dipendenze.
Quindi:
dpkg sta a pacchetti .deb come rpm sta a pacchetti .rpm
apt-get sta a dpkg come yum sta a rpm:

- Elenco completo dei programmi installati sul sistema
rpm -qa (su sistema basati su RPM)
dpkg -l (su sistemi basati su DEB)

- Installare un pacchetto
yum install tcpdump (o rpm -i tcpdump-ver.dist,arch.rpm)
apt-get install tcpdump (o anche dpkg -i tcpdump-ver.arch.deb)

- Rimuovere un pacchetto
rpm -e tcpdump (o anche yum remove tcpdump)
dpkg -r tcpdump (o anche apt-get remove tcpdump)

- Visualizzare l'elenco dei file forniti da un pacchetto
rpm -ql tcpdump (o, se non ancora installato: rpm -qlp tcpdump.arch.rpm)
dpkg -L tcpdump (o, se non ancora installato: dpkg --contents tcpdump-version.arch.deb)

- Visualizzare informazioni su un pacchetto
dpkg -s tcpdump
rpm -qi tcpdump

- Trovare da quale pacchetto è stato installato un dato file
rpm -qf /path/nomefile (es: rpm -qf /etc/issue)
dpkg -S /path/nomefile (es: dpkg -S /etc/issue)

- Aggiornamento del sistema
yum update su sistemi Fedora/RedHat
apt-get upgrade su sistemi Debian e derivati

- Ricerca all'interno del database dei pacchetti
yum search squirrelmail
apt-cache search squirrelmail

Checkinstall Overview, installazione ed uso

Creazione di pacchetti rpm, tgz e deb partendo da una distribuzione sorgente.

OVERVIEW
Checkinstall, scritto da **Felipe Eduardo Sánchez Díaz Durán**, permette di creare partendo da un software distribuito in formato sorgente, tipicamente un file tar.gz, un pacchetto binario per l'installazione su sistemi derivati da **Debian** (.deb), **Slackware** (.tgz) o **Red Hat** (.rpm). Creare un pacchetto nel formato gestito da una determinata distribuzione, comporta il vantaggio di avere un sistema più **ordinato**, in quanto viene mantenuta traccia delle applicazioni correntemente installate, con la possibilità di rimuoverle in modo **pulito**. Cosa che non potrebbe avvenire senza una regola di

uninstall, che spesso manca, nei **makefile** delle applicazioni.

Il processo di installazione di un software mediante checkinstall, varia solamente nell'ultima sua fase. Dopo aver eseguito i classici, `configure` e `make`, checkinstall si sostituisce al `make install`, creando, secondo le esigenze dell'amministratore del sistema, un pacchetto che andrà installato poi con il tool apposito (rpm, apt o installpkg) a seconda della distribuzione di destinazione.

DOWNLOAD

Il download può essere eseguito dalla home del progetto http://asic-linux.com.mx/~izto/checkinstall/:

```
homer@enigma:/software$ wget
http://asic-linux.com.mx/~izto/checkinstall/files/source/checkinst
all-1.6.0beta4.tgz
--13:37:35--http://asic-linux.com.mx/
%7Eizto/checkinstall/files/source/checkin
stall-1.6.0beta4.tgz
=> `checkinstall-1.6.0beta4.tgz'
Resolving asic-linux.com.mx... 200.76.179.225
Connecting to asic-linux.com.mx[200.76.179.225]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147,508 [application/x-tar]
100%
[=============================================================
==========>] 147,508  5.08K/s ETA 00:00
13:38:05 (5.04 KB/s) - `checkinstall-1.6.0beta4.tgz' saved
[147508/147508]
```

INSTALLAZIONE

Una volta scaricati i sorgenti è necessario scompattarli:

```
homer@enigma:/software$ tar xvfz checkinstall-1.6.0beta4.tgz
checkinstall-1.6.0beta4/
checkinstall-1.6.0beta4/FAQ
checkinstall-1.6.0beta4/BUGS
checkinstall-1.6.0beta4/TODO
checkinstall-1.6.0beta4/Makefile
checkinstall-1.6.0beta4/doc-pak/
..
```

quindi eseguire il make:

```
homer@enigma:/software/checkinstall-1.6.0beta4$ make
make -C installwatch-0.7.0beta4
make[1]: Entering directory
`/software/checkinstall-1.6.0beta4/installwatch-0.7.0beta4'
./create-localdecls
```

```
...
make[1]: Leaving directory
`/software/checkinstall-1.6.0beta4/installwatch-0.7.0beta4'
```
ed il make install da root:
```
homer@enigma:/software/checkinstall-1.6.0beta4$ sudo -s
root@enigma:/software/checkinstall-1.6.0beta4# make install
make -C installwatch-0.7.0beta4
make[1]: Entering directory
`/software/checkinstall-1.6.0beta4/installwatch-0.7.0beta4'
...
at:\n\n\t/usr/local/lib/checkinstall/checkinstallrc-dist\n\n======
==================================================\n\n";echo ;fi
```
E' bene sottolineare il fatto che solo in questo caso è necessario utilizzare `make install`, che non installa in realtà il software nel sistema, proprio perchè checkinstall non è ancora operativo.

UTILIZZO

Attraverso l'ultima operazione da compiere per installare checkinstall, è possibile anche vederne l'utilizzo.

Nella cartella in cui sono stati compilati i sorgenti, è necessario a questo punto lanciare il comando `checkinstall`, l'utente sarà quindi guidato tramite una serie di domande:

```
root@enigma:/software/checkinstall-1.6.0beta4# checkinstall

checkinstall 1.6.0beta4, Copyright 2002 Felipe Eduardo Sanchez
Diaz Duran
This software is released under the GNU GPL.

Please choose the packaging method you want to use.
Slackware [S], RPM [R] or Debian [D]? S
```
La prima richiesta del programma riguarda il tipo di pacchetto che si vuole andare a creare. In questo esempio la scelta è S ovvero il formato tgz di Slackware

```
*****************************************
**** Slackware package creation selected ***
*****************************************

This package will be built according to these values:

1 -  Summary: [ CheckInstall installations tracker, version
1.6.0beta4 ]
2 -  Name:    [ checkinstall ]
3 -  Version: [ 1.6.0beta4 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
```

```
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ checkinstall-1.6.0beta4 ]
9 -  Alternate source location: [  ]

Enter a number to change any of them or press ENTER to continue:
4
```
Tramite un menu è possibile selezionare e modificare alcuni valori relativi all'installazione
```
Enter new release number: 4
>> Homer-02-01-2004

This package will be built according to these values:

1 -  Summary: [ CheckInstall installations tracker, version
1.6.0beta4 ]
2 -  Name:    [ checkinstall ]
3 -  Version: [ 1.6.0beta4 ]
4 -  Release: [ Homer-02-01-2004 ]
5 -  License: [ GPL ]
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ checkinstall-1.6.0beta4 ]
9 -  Alternate source location: [  ]

Enter a number to change any of them or press ENTER to continue:

Installing with make install...
========================= Installation results
==========================
make -C installwatch-0.7.0beta4
make[1]: Entering directory
`/software/checkinstall-1.6.0beta4/installwatch-0.7.0beta4'
make[1]: Nothing to be done for `all'.
...

================================================================
===========
An existing checkinstallrc file has been found.
The one from this distribution can be found at:

        /usr/local/lib/checkinstall/checkinstallrc-dist
================================================================
===========
```

```
========================= Installation successful
==========================

Copying documentation directory...
./
FAQ
BUGS
TODO
README
installwatch-0.7.0beta4/
installwatch-0.7.0beta4/BUGS
installwatch-0.7.0beta4/TODO
installwatch-0.7.0beta4/README
installwatch-0.7.0beta4/CHANGELOG
...

Copying files to the temporary directory...OK
Striping ELF binaries and libraries...OK
Compressing man pages...OK
Building file list...OK
Preparing Slackware install directory...OK
Writing package description...OK
Creating package
checkinstall-1.6.0beta4-i386-Homer-02-01-2004...OK
NOTE: The package will not be installed.
Erasing temporary files...OK
Writing backup package...OK
Deleting temp dir...OK

*****************************************************************
****
Done. The new package has been saved to

/software/checkinstall-1.6.0beta4/checkinstall-1.6.0beta4-i386-Hom
er-02-01-2004.tgz
You can install it in your system anytime using:

      installpkg
checkinstall-1.6.0beta4-i386-Homer-02-01-2004.tgz
*****************************************************************
****
```

Una volta in possesso di tutte le informazioni necessarie, checkinstall crea il pacchetto per la distribuzione scelta ed

informa l'utente della sua locazione e di come fare per installarlo

Una volta creato, il pacchetto è pronto per l'installazione:

```
root@enigma:/software/checkinstall-1.6.0beta4# installpkg
checkinstall-1.6.0beta4-i386-Homer-02-01-2004.tgz
Installing package
checkinstall-1.6.0beta4-i386-Homer-02-01-2004...
PACKAGE DESCRIPTION:
Executing install script for
checkinstall-1.6.0beta4-i386-Homer-02-01-2004...
```

DESCRIZIONE DEL PACCHETTO

E' possibile creare una descrizione del pacchetto che si andrà ad installare inserendo le informazioni in un file chiamato `description-pak`. Qualora questo file risulti assente, Checkinstall provvederà ad inoltrare una richiesta all'utente per procedere ad una sua eventuale creazione. Le informazioni inserite verrano visualizzate durante l'utilizzo di `pkgtool`, in Slackware oppure tramite il comando `rpm -qi` nelle distribuzioni rpm based.

SCRIPT PRE E POST INSTALLAZIONE/RIMOZIONE

Per i package di tipo .rpm o .deb, è possibile creare a seconda dell'occorrenza quattro tipi di **shell script**, che devono essere presenti nella directory dei sorgenti:
preinstall-pak: eseguito prima dell'installazione;
postinstall-pak: eseguito dopo l'installazione;
preremove-pak: eseguito prima delle rimozione;
postremove-pak: eseguito dopo la rimozione;
questi tipi di script possono risultare utili, nell'**automatizzare** operazioni quali per esempio la creazione o cancellazioni di utenti, in relazione con il software che si andrà ad installare o rimuovere.

rpm --test [ivh | Uvh] [pacchetto.rpm]

Permette di verificare eventuali problemi prima dell'installazione di un pacchetto.

Prima di installare, o aggiornare un pacchetto su un sistema, è opportuno verificare che non ci siano problemi di dipendenze o di librerie. E' possibile varo con utilizzando il flag `--test` di rpm:

```
[azitti@fantasy azitti]$ rpm --test -Uvh cvs-1.11.17-1.i386.rpm
error: failed dependencies:
        libc.so.6(GLIBC_2.3)   is needed by cvs-1.11.17-1
```

In questo caso è possibile notare che rpm segnala la dipendenza dalla libreria libc.so.6 non presente

rpm -qf `which [executable]`

Utilizzo della "command substitution" per conoscere di un eseguibile il pachetto che lo conteneva e

la relativa versione.

```
rpm -qf `which tr`
```

*voglio sapere il comando **tr** a quale pacchetto appartiene*

```
coreutils-4.5.3-19.0.2
```

*la risposta è **coreutil** alla versione 4.5.3*

rpm -qa --qf "%-10{SIZE} %-30{NAME}\n" | sort -n

Con questa semplice riga è possibile visualizzare l'elenco dei pacchetti RPM installati sul sistema ordinandoli secondo le dimensioni.
Molto comodo per vedere quali pacchetti occupano più spazio.

Fonte: FedoraNews. Autore: Leonid Mamtchenkov

Import delle chiavi GPG per installare i pacchetti rpm

I pacchetti rpm che venongo distribuiti da chi produce una distribuzione o da chi gestisce repository alternativi, sono quasi sermpre "firmati" con una chiave GPG che certifica l'autore del pacchetto, assicurandone la fonte.
E' bene, al termine di una installazione Linux, importare le chiavi pubbliche (`rpm --import`) dei packager che forniscono gli aggiornamenti o i pacchetti che si intendono utilizzare. Questa operazione potrebbe rendersi indispensabile quando si utilizzano tool di aggiornamento automatici come yum o autorpm se sono configurati per eseguire il gpgcheck dei pacchetti da aggiornare.

Se si prova ad installare un pacchetto creato da un packager di cui non si è importata la chiave GPG pubblica si ottiene un output di questo genere:

```
[root@zoe root]# rpm -Uhv httpd-2.0.40-21.3.i386.rpm
warning: httpd-2.0.40-21.3.i386.rpm: V3 DSA signature: NOKEY, key
ID db42a60e
Preparing...                ########################################
##### [100%]

1:httpd                     ########################################
## [100%]
```

Le chiavi GPG pubbliche sono comunemente rintracciabili nella directory principale del primo CD di installazione o in /usr/share/doc/, ad esempio su Fedora in /usr/share/doc/fedora-release-XX/RPM-GPG-KEY:

```
[root@zoe root]# rpm --import
/usr/share/doc/fedora-release-2/RPM-GPG-KEY
[root@zoe root]# rpm -Uhv httpd-manual-2.0.40-21.3.i386.rpm
Preparing...                ########################################
##### [100%]
   1:httpd-manual
```

```
########################################### [100%]
```

Conversione di RPM in tgz

Tramite l'utility rpm2tgz è possibile trasformare un pacchetto RPM nel formato gestito dal Package Management di Slackware

Uno dei formati più popolari tra i sistemi di packaging è RPM di Red Hat, ed è quindi probabile trovare spesso software in questo formato. A volte è possibile che il pacchetto ricercato sia disponibile solamente in questo formato. In questo caso è possibile trovare una soluzione (anche se non in tutti i casi), per l'installazione in una distribuzione Slackware, grazie all'utility **rpm2tgz**.

Il programma rpm2tgz crea un pacchetto con estensione .tgz partendo da un pacchetto RPM, la sua sintassi è: `rpm2tgz <file.rpm>`

Un esempio di utilizzo:

```
root@Joker:/test# ls
nmap-3.00-1.i386.rpm
Il file di partenza è un pacchetto di tipo RPM
root@Joker:/test# rpm2tgz nmap-3.00-1.i386.rpm
root@Joker:/test# ls
nmap-3.00-1.i386.rpm  nmap-3.00-1.i386.tgz
Una volta lanciata l'esecuzione del programma si ottiene un
pacchetto di tipo tgz gestibile dalle utility di Slackware come
pkgtool, installpkg, removepkg o upgpackage
```

Slackware Package Management

Anche la distribuzione Linux Slackware utilizza software di Package Management, che sebbene più spartano rispetto a RPM di Red Hat o a Deb di Debian, permette di mantenere in ordine i pacchetti installati nel sistema

I packages di Slackware sono dei semplici tar compressi con gzip. Questa distribuzione mette a disposizione alcune utility per gestirne l'installazione, la rimozione, l'aggiornamento e di mantenere traccia delle operazioni tramite un database.

I principali tools forniti da Slackware per il management delle applicazioni sono quattro: **pkgtool, installpkg, removepkg upgradepkg**.

pkgtool
Pkgtool è un'utility di tipo menu-driven che permette di visualizzare, installare e rimuovere i packages. Attraverso questo tool è possibile visualizzare il contenuto di ogni package, disinstallarlo o scegliere da dove installarne uno nuovo. Tramite pkgtool non è però possibile effettuare l'aggiornamento, prerogativa disponibile solo per le utility di tipo command line che dispongono anche di un maggior numero di opzioni.

Passiamo agli strumenti a linea di comando.

installpkg
Installpkg gestisce l'installazione di nuovi packages nel sistema.
Sintassi:
`root@Joker:/# [ROOT=<path>] installpkg [options] <nome package>...`
Opzioni:
`-m`: Esegue un `makepkg` (*Utility per creare i packages*) nella directory corrente;
`-warn`: Visualizza i cambiamenti nel sistema in caso di installazione del package. Usato sulle macchine di produzione per sapere cosa accadrà installando un software;
`-r`: Installa *ricorsivamente* i packages contenuti nella directory corrente e nelle subdirectory. E' possibile utilizzare delle wildcards.

Settando la variabile *ROOT* è possibile utilizzare una directory a propria scelta, diversa da `/`, per memorizzare i dati relativi all'installazione.

Al termine dell'installazione, se presente, nella subdirectory `install/` del package verrà eseguito uno script di nome `doinst.sh` che permette di rifinire l'installazione creando per esempio link simbolici.

Le informazioni del database dei packages installati, ovvero un file in plain text per ogni programma, si trovano in `/var/log/packages` mentre gli eventuali script di post installazione si trovano in `/var/log/scripts/<nomepackage>`.

removepkg
Removepkg si occupa di disinstallare i packages dal sistema.
Sintassi:
`root@Joker:/# [ROOT=<path>] removepkg [options] <nome package>...`
Opzioni:
`-copy`: Il package non viene rimosso ma viene copiato in in una directory in `/var/log/setup/tmp/preserved_packages` uguale all'originale;
`-keep`: Tiene traccia dei file temporanei creati durante la disinstallazione. E' comodo per scopi di debugging;
`-preserve`: Il package viene rimosso, ma copiato per sicurezza in un'altra directory, ovvero `/var/log/setup/tmp/preserved_packages`;
`-warn`: Visualizza quali problemi potrebbero esserci rimuovendo il package;

Settando la variabile *ROOT* è possibile utilizzare una directory a propria discrezione, diversa da `/`, per memorizzare le informazioni relative alla disinstallazione.

Se presente, removepkg esegue lo script di postinstallazioine in modo da rimuovere, oltre ai file del package, anche eventuali link simbolici presenti.

Durante il processo di disinstallazione, vengono visualizzate le informazioni sullo stato dell'operazione. Una volta terminato li processo, le informazioni del packages e lo script di post installazione sono rispettivamente spostati in `/var/log/removed_packages` e `/var/log/removed_scripts`.

upgradepkg
Upgradepkg gestisce l'aggiornamento di un package Slackware già installato.

`root@Joker:/# [ROOT=<path>] upgradepkg <package name>...`
oppure:
`root@Joker:/# [ROOT=<path>] upgradepkg [options] <vecchio nome package> <nuovo nome package>`

Upgradepkg esegue nell'ordine, l'installazione del nuovo package e la disinstallazione del vecchio. Se il nome del package è cambiato da una versione all'altra è possibile usare la seconda versione del comando.

Al fine di evitare i problemi dovuti a qualche bug di upgradepkg come per esempio la sovrascrittura accidentale dei file di configurazione, è sempre consigliato fare un backup dei propri file di configurazione.

In tutte e tre le utility è possibile specificare più di un package utilizzando nel nome le wildcards.

rpm -qa

Visualizza l'elenco completo degli rpm installati.
Utile sopratutto se usato assieme al grep.

Nel caso volessi sapere se il pacchetto abiword è **installato** sul mio sistema, dovrei digitare al prompt:
```
[root@hell] rpm -qa | grep abiw
```

Patch e aggiornamento di Sun Solaris

Come ogni sistema operativo Solaris viene costantemente aggiornato, sia per la scoperta di bug o buchi di sicurezza nella versione rilasciata inizalmente al pubblico, sia per l'aggiunta di nuove funzionalità o il suopporto di nuovo hardware. Gli aggiornamenti di Solaris vengono fatti tramite "patch" che possono essere scaricati direttamente dal sito di Sun.

Il punto di riferimento per la verifica delle patch per il proprio sistema e l'aggiornamento è `http://sunsolve.sun.com`. Qui si possono browsare e ricercare le patch e scaricare sia quelle "Reccomended" (download aperto a tutti" sia tutte le altre (download riservato ai clienti che hanno un contratto di assistenza con Sun). Lo storico delle patch installate sul sistema sta in

`/var/sadm/patch`, questa directory non va cancellata in quanto permette l'aggiunta o rimozione di patch.
Da Solaris 7 in poi le patch sono in un file .zip, con un formato tipo: **112438-01.zip**. Il primo numero, di 6 cifre, è un ID univoco diverso per ogni patch. Il secondo numero, di 2 cifre, è la revisione per quella patch. E' un numero progressivo, revisioni successive (es: 10) comprendono tutte le precedenti (es: 07).

I comandi utili per gestire le patch sono `patchadd`, `patchrm` e `showrev`.

Una volta unzippata la patch (es: `unzip 112438-01.zip`, si ottiene una directory, col nome della patch, che contiene vari file e sottodirectory con questa logica:
`README.112438` Instruzioni per l'installazione
`postpatch` Script eseguito al termine del processo di patch
`postbackout` Script da eseguire in caso di rimozione della patch
`SUNWmdbx/` Directory con il nome pacchetto che viene aggiornato (possono essercene più di una per patch).
`SUNWmdbx/pkgmap` Path dei nuovi file o directory nella patch
`SUNWmdbx/pkginfo` Informazioni aggiornate per il pacchetto (vanno in `/var/sadm/pkg/SUNmdbx/pkginfo`)
`SUNWmdbx/install/` Directory che contiene script da eseguire durante il patching
`SUNWmdbx/reloc/` Directory che contiene i nuovi file che vanno a sostituire quelli esistenti (con PATH completo)

Per installare la patch basta un comodo (da Solaris 7 in poi):
`patchadd 112438-01` che esegue il comando `pkgadd` e gli script presenti nella patch.
Con l'aggiornamento vengono eseguite le seguenti modifiche in `/var/sadm`:
- Viene aggiunta la directory 112438-01 in `/var/sadm/patch`;
- Viene aggiornato il file `/var/sadm/pkg/PACCHETTIAGGIORNATI/pkginfo` (ovviamente per PACCHETTIAGGIORNATI si intendoni nomi come SUNWmdbx)
- Viene creata la directory `/var/sadm/pkg/PACCHETTIAGGIORNATI/save` dove ci sono informazioni e binari per il ripristino dei vecchi file.

E' possibile ripristinare la situazione pre-patch con il comando `patchrm 112438-01` (possibile solo se la patch non è richiesta (required) da un'altra patch o se non è stata resa obsoleta (obsoleted) da un ulteriore patch)

Con il comando `showrev -p` (o `patchadd -p` è possibile visualizzare l'elenco delle patch installate.

Sun, inoltre, su Sunsolve.sun.com, mette a disposizione dei Cluster di Patch costantemnte aggiornati, che contengono tutte le patch reccomended e permettono di eseguire in un unica operazione l'aggiornamento completo del sistema operativo. I cluster di patch sono paragonabili ai

Service Packs di Microsoft, con la differenza che vengono costantemente aggiornati.

Per installare un cluster di solito basta unzippare l'enorme file, entrare nella directory ottenuta ed eseguire `./install_cluster` (sempre meglio leggersi i README del caso, prima).

Gestire pacchetti software con Solaris: pkgadd e altri comandi

Solaris utilizza un sistema di gestione dei programmi in pacchetti analogo agli RPM o DEB comuni in Linux.
Su Solaris si parla di packages, .pkg, che contengono binari già compilati, manuali, file di configurazione e generalmente tutto quello che serve per installare un software sul sistema.
Gli strumenti per gestire i packages sono vari:
- Command line (comandi pkgadd, pkgrm, pkginfo, pkgchk, pkgask);
- Admintool, strumento grafico disponibile sotto Xwindows;
- Solaris Product Registry, altro strumento grafico, disponibile da Solaris 8.

Questi strumenti sono intercambiabili e convivere senza problemi: un pacchetto può, per esempio, essere installato con Admintool e rimosso com pkgrm.

Occupiamoci degli strumenti in command line, una volta capita la loro logica, diventa triviale utilizzare le alternative grafiche.

`/bin/pkgadd`
Installa uno o più pacchetti. Utilizza di default i comportamenti definiti nel file di configurazione `/var/sadm/install/admin/default` in cui sono presenti varie coppie di parametri=valore che definiscono come gestire l'installazione di pacchetti a seconda delle situazioni (se chiedere all'utente, procedere automaticamente, annullare l'operazione ecc in presenza di determiante situazioni: pacchetto già installato, completamente o parzialmente, spazio su disco insufficente, conflitti con altri pacchetti, necessità di dependencies, run level, controllo sull'impatto sulla sicurezza, directory di installazione ecc).
Il file di default, in genere, chiede all'utente come procedere quando incontra una delle suddette impostazioni ed ha come directory predefinita `/opt`.
La sintassi di pkgadd:
`-d nome_device` Opzione facoltativa. Specifica il device (es: /cdrom/cdrom0) in cui si trova(no) i(l) pacchetto(i).
`-a file_di amministrazione` Opzione facoltativa, permette di specificare un file di configurazione/amministrazone diverso da /var/sadm/install/admin/default. Se si specifica solo il nome, senza path, viene cercato in /var/sadm/install/admin/.
`nome_pacchetto` Il nome (pkgid) di uno o più pacchetti che si voglinno installare. Se non specifica nessun pacchetto e nessun device, pkgadd cerca se ce ne sono nella directory di spool `/var/spool/pkg` e li installa automaticamente.

`/usr/sbin/pkgchk`
Verifica l'integrità di un pacchetto. Va fatto seguire dal nome del pacchetto (pkgid) e se non da nessun output se il pacchetto è installato correttamente.

Se non si specifica alcun nome di pacchetto, esegue il check di tutti i pacchetti installati sul sistema.
`-d directory_di_spool` Controlla i pacchetti nella directory di spool (o device) specificati. Le informazioni non sono complete, comunque, fino a quando il pacchetto non viene effettivamente installato.
`-v` Elenca tutti i file contenuti nel pacchetto. Questi dati vengono registrati nel file `/var/sadm/install/contents`.
Per verificare in quale pacchetto è contenuto un determinato file digitare:

```
pkgchk -l -p nomefile
```

`/bin/pkgparam`
Visualizza i parametri del pacchetto specificato.
`-v` Opzione per visualizzare sia il nome del parametro che il suo valore (di default visualizza solo il valore).

`/bin/pkginfo`
Visualizza tutti i pacchetti installati sul sistema con una breve descrizione.
`-d directory` Visualizza l'elenco dei pacchetti contenuti sulla directory o device specificati.

`/usr/sbin/pkgrm`
Rimuove il pacchetto specificato.
`-n` Rimuove il pacchetto in modalità non interattiva
`-s directory_di_spool` Rimuove il pacchetto (se non viene speicificato li rimuove tutti) dalla directory di spool indicata.