# 7. ADMINISTRAREA SOFTWARE-ULUI

## 7.1. Forme ale software-ului si caracteristicile acestora

Printre responsabilitatile administratorului de sistem se numara si operatiile de instalare, dezinstalare si mentenanta a software-ului, fie el parte a sistemului de operare sau aplicatie separata.

Software-ul poate fi gasit sub doua forme:

- **sursele programului** – disponibile numai in cazul softului free sau open source. Pentru a obtine forma finala (executabila) a unui astfel de soft este necesara compilarea si instalarea lui de catre utilizator sau administratorul de sistem
- **pachete precompilate ("binary packages")** – contin forma deja compilata a softului (fisiere executabile, biblioteci de functii si alte resurse). Sunt folosite pe scara larga si in cazul sistemelor de operare open source, datorita usurintei administrarii (vezi sectiunea despre pachete)

Nici una dintre cele doua forme nu ii este superioara celeilalte decat punctual, in functie de scenariu. Iata principalele caracteristici ale celor doua, urmand ca ele sa fie detaliate in continuarea materialului:

- **pachete precompilate**
  - avantaje
    - *timpul necesar operatiilor de instalare/upgrade*. Deoarece pachetele contin soft deja compilat, singurele operatii necesare la instalare sunt copierea fisierelor in locatiile corespunzatoare si eventual generarea/actualizarea fisierelor de configurare
    - *urmarirea interdependentelor intre pachete*. Precum se va vedea mai jos, este posibil ca o aplicatie sa fie distribuita sub forma mai multor pachete, si in plus sa depinda de multe altele pentru a putea functiona
    - *mentinerea unei baze de date cu pachetele instalate.* Baza de date in cauza nu contine numai lista de pachete, ci si numele/caile fisierelor continute si dependintele pachetului. Astfel, in orice moment putem sti ce software am instalat in sistem, ce resurse a instalat un anume pachet, de ce pachet apartine un anumit fisier de pe hard-disk etc.
  - dezavantaje
    - compilarea a fost efectuata cu setari prestabilite, care nu mai pot fi modificate ulterior
    - pachetul nu este in general perfect optimizat pentru arhitectura de procesor pe care se instaleaza. Deoarece creatorul pachetului nu poate sti de la bun inceput pe ce platforma va fi instalat acesta, el este nevoit sa foloseasca setari care sa asigure compatibilitate maxima cu o gama cat mai larga de platforme
- **surse**
  - avantaje:
    - *flexibilitate*. Administratorul cu cunostinte de programare poate modifica codul sursa dupa necesitati, si chiar si cel lipsit de astfel de competente are la dispozitie modalitati de a efectua setari ce influenteaza forma finala a soft-ului compilat (module sau facilitati ce vor fi sau nu incluse, optimizari, locatia de instalare etc)
  - dezavantaje
    - *timp necesar*. Pentru a obtine forma executabila a softului, sursele trebuie sa treaca prin procesul de compilare care, in functie de anvergura aplicatiei compilate, poate fi mare consumator de timp. Abia dupa efectuarea compilarii se trece la etapa de instalare (singura existenta in cazul pachetelor precompilate)
    - *nu exista urmarirea automata a software-ului instalat*. O aplicatie compilata si instalata din surse nu se inregistreaza in nici o baza de date, astfel incat nu putem obtine usor lista de aplicatii instalate la un moment dat in sistem
    - d*ezinstalarea poate fi dificila*. Atunci cand, la instalarea unui soft, componentele sale se amesteca cu cele ale altor aplicatii (ex: in /usr/bin rezida excutabile ale diverselor

programe instalate in /usr), si daca acel soft nu pune la dispozitie o modalitate de dezinstalare, a sterge acel soft ar presupune identificarea tuturor fisierelor ce apartin de acel soft si stergerea lor manuala

- *competente necesare*. Daca, din varii motive, compilarea esueaza, utilizatorul trebuie sa determine cauza analizand mesajele brute furnizate de compilator, care sunt criptice pentru neinitiati

## *7.2. Pachete precompilate*

### 7.2.1. Conceptul de pachet precompilat

Un pachet precompilat ("binary package") este pana la un punct asemanator ca functionalitate cu kitul de instalare din Windows: el contine elementele componente ale programului ce se doreste instalat (fisiere executabile, biblioteci de functii, fisiere de configurare etc), urmand ca, in cadrul procesului de instalare, acestea sa fie copiate in locatiile prestabilite din sistemul de fisiere.

Un pachet precompilat contine:

- ierarhia de fisiere si directoare ce se vor copia pe hard-disk (executabile, biblioteci, fisiere de configurare, imagini, documentatie etc) - cu caile lor absolute!
- informatii despre fiecare fisier in parte (owner, group, permisiuni, suma de control)
- informatii despre pachet (descriere, elemente pentru verificarea autenticitatii si integritatii, lista de dependinte)
- scripturi care controleaza intregul proces de instalare

Dependintele unui pachet reprezinta lista de pachete de care el depinde pentru a se putea instala - cele indispensabile functionarii acestuia. Acesta este unul dintre aspectele ce diferentiaza pachetul Linux de kitul Windows: daca in Windows, in cele mai dese cazuri, un kit contine toate cele necesare pentru rularea aplicatiei instalate, in Linux o aplicatie este in general divizata in mai multe pachete interdependente, si in plus este posibil ca pachetele componente sa depinda la randul lor de altele pentru a putea functiona. In aceste conditii, deseori instalarea unui singur pachet nu este suficienta: pentru a putea rula softul dorit este necesara instalarea tuturor dependintelor pachetului in cauza. Spre exemplu, in Ubuntu, Chromium BSU (un joc) este format din pachetul *chromium-bsu* si *chromium-bsu-data*, care la randul lor au o intreaga lista de dependinte.

### 7.2.2. Tipuri de pachete

Exista numeroase tipuri de astfel de pachete, fiecare cu propriul format si capabilitati; iata-le pe cele mai importante dintre ele, fiecare provenind de la cate o distributie “clasica” de Linux:

- **.rpm** (Redhat Package Manager) – tipul de pachete lansat in RedHat Linux si prezent in distributii derivate din acesta (Fedora, CentOS etc) sau independente (ex: OpenSuse)
- **.deb** – formatul de pachet caracteristic distributiei Debian si derivatelor acesteia (Ubuntu, Mint etc.)
- **.tgz** sau **.txz** – formate de pachet arhivat si comprimat, intalnite in Linux Slackware si derivatele acestuia
- **.pkg.tar.xz** - formatul de pachet din Arch Linux

***Nota:*** *au fost create si utilitare de conversie dintr-un tip de pachet intr-altul. Iata cateva exemple:*

- *rpm2tgz sau rpm2targz – utilitare ce convertesc pachete rpm in formatul Slackware (tgz)*
- *alien – utilitar de conversie intre rpm, dpkg si tgz*

## 7.2.3. Conceptul de manager de pachete si avantajele sale

Un manager de pachete este programul prin intermediul caruia se efectueaza toate operatiile de administrare de soft dintr-o distributie Linux. Iata cateva dintre facilitatile oferite de acesta:

- realizeaza instalarea pachetelor, conform instructiunilor cuprinse in pachet
- determina dependintele pachetelor ce se doresc instalate si avertizeaza utilizatorul in privinta lor
- mentine o baza de date cu software-ul instalat, ce permite determinarea in orice moment a listei de pachete instalate si a fisierelor provenite din fiecare pachet
- efectueaza dezinstalarea controlata a pachetelor, avertizand in privinta dependintelor (atunci cand dorim sa stergem un pachet, este posibil sa existe altele, deja instalate, care depind de el)
- permite upgrade-ul (aducerea la o versiune mai noua) a unui pachet deja instalat
- permite verificarea integritatii fisierelor de pe hard disk ce apartin de un pachet

***Remarca:*** *o alta diferenta fata de sistemul de management al software-ului din Windows este ca acolo multe softuri aveau propriul lor installer, instalarea si dezinstalarea efectuandu-se prin intermediul installerului inclus in kit, iar Windows-ul inregistrand doar softul instalat in baza sa de date. In Linux avem o aplicatie centralizata care realizeaza aceste operatii (managerul de pachete), pachetul necontinand codul care realizeaza instalarea.*

## 7.2.4. Lucrul cu pachete RPM

### *7.2.4.1. Descriere si caracteristici*

RPM este un format de pachet creat de catre RedHat Linux si prezent astazi pe o multitudine de alte distributii (Fedora, Suse, CentOS etc). Chiar si in cazul distributiilor care nu folosesc rpm ca format nativ de pachete, exista posibilitatea lucrului cu pachete rpm – fie prin utilitare de conversie la alt format de pachet, fie prin instalarea sistemului rpm pe distributia in cauza.

***Atentie!*** *Desi multe distributii folosesc formatul RPM, iar utilitarele si sintaxa de lucru cu pachete sunt identice, compatibilitatea este departe de a fi perfecta intre distributii. Incercati pe cat posibil sa folositi pachete RPM create special pentru distributia pe care lucrati.*

Denumirea unui pachet RPM este de forma:

```
nume-versiune-release.arhitectura.rpm
```

unde elementele componente au urmatoarele semnificatii:

- **nume** – reprezinta numele aplicatiei/bibliotecii continute in pachet. Aceasta nu inseamna ca pachetul este suficient pentru functionarea aplicatiei in cauza – este posibil ca el sa aiba dependinte.
- **versiune** – indica versiunea softului inclus in pachet. Nu putem avea instalate doua versiuni diferite ale aceluiasi soft decat in masura in care numele de pachet difera. De aceea, atunci cand avem totusi nevoie de versiuni diferite, distributiile recurg la o stratagema: includ versiunea in numele de pachet. Spre exemplu, putem intalni pachete precum libglib1.2 si libglib2.0, ce corespund versiunilor 1.2 si 2.0 ale aceluiasi soft – glib.
- **release** – folosit pentru a indica versiunea de pachet si deseori distributia pentru care a fost generat. Pe baza aceleiasi versiuni a unui soft pot fi create diferite versiuni de pachet ("builds"). In plus, pentru o aceeasi versiune a aceluiasi soft exista in general pachete pentru diverse distributii (vezi exemple)
- **arhitectura** – se refera la clasa de procesoare al carei set de instructiuni a fost folosit pentru generarea codului executabil cuprins in pachet. Implica doua aspecte:

- **arhitectura de procesor** pentru care a fost generat codul (Intel, SPARC, PowerPC etc). Procesoarele "inteleg" succesiuni de numere, ce reprezinta pentru ele instructiuni de executat, insa pentru diferite arhitecturi de procesor numerele vor avea semnificatii diferite. De aceea un soft compilat pentru procesoare Intel nu va putea rula pe un procesor SPARC.

***Nota:*** *si procesoarele AMD se incadreaza tot in arhitectura Intel! (x86 sau x86_64)*

- **generatia de procesor** din cadrul arhitecturii – de exemplu, in cadrul arhitecturii Intel (cea mai raspandita), procesorul initial a fost 8086, urmand 80186, 80286, 80386 etc. Fiecare noua generatie a introdus instructiuni si capabilitati noi, ramanand insa compatibila cu generatiile anterioare (spre exemplu, codul scris pentru un 80386 va rula si pe procesoare Pentium sau mai noi). Codul executabil ce foloseste instructiuni ale unei generatii mai noi va putea avea un spor de viteza spectaculos, insa va deveni inaccesibil utilizatorilor cu procesoare mai vechi; acesta este motivul pentru care pachetele RPM tintesc deseori arhitecturi joase (386, i586 etc), care ofera compatibilitate cu o gama foarte larga de procesoare din arhitectura Intel

***Nota:*** *pachetele RPM cu arhitectura* noarch *sunt independente de arhitectura procesorului (ex: pachete care contin doar scripturi, imagini, documentatie etc)*

| Arhitectura/ generatie | Procesoare |
|---|---|
| i386 | 80386 |
| i586 | Pentium, Pentium MMX |
| i686 | Pentium Pro/II/III/IV/M, AMD K5/K6/Athlon |
| x86_64 | Athlon64, Turion, Intel Core 2 si superioare |
| ppc | PowerPC |

Exemple:

```
# openssl versiunea 1.0.2d, versiune de pachet 2, instructiuni x86_64, distributie Fedora core 24
openssl-1.0.2d-2.fc24.x86_64.rpm

# pidgin versiunea 2.10.11, versiune de pachet 5, pt arhitectura intel 586, distributie Mageia 6
pidgin-2.10.11-5.mga6.i586.rpm

# acelasi soft si versiune, versiune de pachet 12, pentru Fedora Core 22
pidgin-2.10.11-12.fc22.i686.rpm
```

Administrarea pachetelor RPM se face cu ajutorul comenzii **rpm** (de la Redhat Package Manager), care are o sumedenie de optiuni, corespunzatoare unui numar de operatiuni de baza: instalare, verificare, interogare, stergere, upgrade etc. Utilitarul mentine o baza de date cu pachetele instalate, oferind posibilitatea determinarii in orice moment a setului de pachete prezente in sistem. **rpm** stie sa atentioneze atunci cand un pachet ce se doreste instalat are nevoie de alte pachete neinstalate inca, si inglobeaza un sistem de testare a autenticitatii si validitatii pachetelor. Baza de date se gaseste in */var/lib/rpm/*.

Prezentam in continuare principalele operatiuni cu pachete RPM.

### 7.2.4.2. Instalare

Instalarea unui pachet se realizeaza cu optiunea **–i** a comenzii **rpm**. Pachetul poate proveni din sistemul de fisiere local sau din retea, **rpm** putand lucra cu URL-uri ce trimit catre servere FTP sau HTTP:

```
rpm –ivh /tmp/openssl-1.0.2d-2.fc24.x86_64.rpm
rpm –ivh ftp://rpmfind.net/linux/fedora/linux/development/rawhide/i386/os/Packages/o/openssl-
1.0.2d-2.fc24.i686.rpm
```

Optiunea **–v** (verbose) determina un output mai detaliat al comenzii, iar **–h** (hash) adauga un *progress indicator* format din caractere #.

Atunci cand pachetul dorit are dependinte care nu sunt inca instalate in sistem, exista trei rezolvari:

1. Se obtine lista de fisiere necesare, folosind comanda **rpm –qRp pachetcudependinte.rpm**, si se obtin&instaleaza acestea separat (necesita un grad ridicat de „indemanare administrativa” si de rabdare)
2. Daca suntem siguri ca dependintele nu vor fi necesare, putem instala pachetul fortat, ignorand dependintele, cu comanda **rpm --nodeps pachetcudependinte.rpm**. Riscul este insa ca softul instalat sa nu porneasca sau sa aiba functionalitati lipsa
3. Folosim un integrator de pachete, care ofera serviciul detectarii, obtinerii si instalarii automate a dependintelor pachetului dorit

Inca cateva comenzi utile pentru aceasta operatiune puteti gasi mai jos, in cadrul sectiunii „Interogare”.

***Nota:*** *exista cazuri (rare) de dependinte ciclice: pachetul 1 are nevoie de 2, 2 are nevoie de 3, iar 3 de 1! Solutia este specificarea tuturor celor 3 pachete in linia de comanda rpm ce realizeaza instalarea:* ***rpm –ivh pachet1.rpm pachet2.rpm pachet3.rpm***

### 7.2.4.3. Dezinstalare

Optiunea rpm folosita pentru dezinstalare este **–e** (erase). Argumentul primit nu trebuie sa fie denumirea completa a pachetului, ci este suficient numele acestuia:

```
rpm –ev nume_soft
```

Inaintea dezinstalarii, managerul de pachete verifica daca exista alte pachete care depind de cel specificat, in caz afirmativ nepermitand stergerea acestuia.

***Atentie!*** *Daca pentru instalarea unui pachet a fost necesara instalarea unei serii de pachete-dependinta, la stergerea aceluiasi pachet nu vor fi sterse automat si pachetele in cauza! Facilitatea de stergere automata a dependintelor ce nu mai sunt folosite se regaseste de obicei in softurile de tip integrator de pachete (APT, yum, dnf, zypper etc)*

### 7.2.4.4. Interogare

Managerul de pachete *rpm* mentine o baza de date bogata in informatii despre pachetele instalate in sistem si fisierele acestora. Interogarile pe care el le pemite pot fi adresate fie acestei baze de date, fie unui pachet rpm neinstalat inca si despre care se doresc informatii. Putem afla:

- toate pachetele instalate: **rpm –qa**. Utilizare uzuala: **rpm –qa|grep *nume_soft***
- este pachetul *nume* instalat? **rpm –q *nume***
- din ce pachet face parte un anume fisier din sistemul de fisiere: **rpm –qf /etc/passwd** (cale absoluta!)
- informatii despre un pachet instalat: **rpm –qi *nume_soft***
- informatii despre un pachet neinstalat: **rpm –qip /cale/catre/pachet.rpm**
- lista de fisiere dintr-un pachet instalat: **rpm –ql nume | less**
- lista de fisiere dintr-un pachet neinstalat: **rpm –qlp /cale/catre.pachet.rpm | less**
- dependintele unui pachet pe care dorim sa-l instalam: **rpm –qRp /cale/catre/pachet.rpm**
- pachetele ce depind de *nume_pachet*: **rpm -q --whatrequires *nume_pachet***

### 7.2.4.5. Upgrade

Exista doua posibilitati de a inlocui pachetele deja existente cu versiuni mai noi:

- **upgrade** – daca pachetul exista deja, este actualizat la noua versiune; daca nu exista, va fi instalat**:**

```
rpm –Uvh /cale/catre/pachet.rpm
```

- **freshen** – sunt actualizate numai acele pachete care sunt deja instalate:

```
rpm –Fvh ftp://ftp.linux.ro/centos/7/os/x86_64/Packages/ImageMagick-devel-6.7.8.9-10.el7.i686.rpm
```

#### *7.2.4.6. Verificare*

Putem face doua tipuri de verificari:
1. Compararea fisierelor instalate ale unui pachet cu informatia prezenta in baza de date: **rpm –V *numepachet***
2. Verificarea semnaturii digitale si a sumei de control: **rpm –K *nume*** sau **rpm --checksig *nume***.

Verificarea autenticitatii si integritatii pachetului necesita importarea anterioara a cheii publice a distribuitorului pachetului, cu comanda **rpm --import */cale/catre/cheie***. Cheile sunt administrate ca si pachetele – pot fi vizualizate cu **rpm -qa gpg-pubkey***, si pot fi sterse cu **rpm –e *nume_cheie***. Informatii despre o cheie se obtin cu **rpm –qi *nume_cheie***.

### 7.2.5. Administrarea pachetelor .deb (Debian)

Sub Debian exista o suita de utilitare ce pot realiza aproximativ aceleasi operatii ca si *rpm*:
- instalare: **dpkg -i */cale/catre/pachet.deb***
- dezinstalare: **dpkg -r */cale/catre/pachet.deb***
- reconfigurare pachet: **dpkg-reconfigure *nume_pachet***
- afisare informatii despre pachet: **dpkg -p *nume_pachet***
- listare pachete instalare: **dpkg -l**
- listare fisiere dintr-un pachet instalat: **dpkg -L *numepachet* | less**

In general exista front-end-uri pentru dpkg:
- **dselect** – un utilitar bazat pe meniuri in mod text
- **apt** (Advanced Package Tool - portat ulterior si pe alte distributii) – un ansamblu de utilitare de linie de comanda care automatizeaza lucrul cu pachete. Ca si rpm, APT a fost ulterior portat si pe alte distributii. APT include comenzi precum **apt-get** (ce permite instalarea sau stergerea pachetelor), **apt-cache** (ce permite cautarea in lista de pachete disponibile) etc. - vezi sectiunea despre integratoare de pachete
- **aptitude** – interfata in mod text pentru apt-get. Este bazat pe meniuri si foarte usor de folosit pentru cei care nu doresc sa retina optiunile comenzilor componente ale APT

### 7.2.6. Administrarea pachetelor .tgz (Slackware)

Administrarea software-ului sub Linux Slackware se face cu comenzile **pkgtool, installpkg, removepkg** si **upgradepkg**. Prima dintre acestea este o aplicatie text-based ce utilizeaza meniuri si integreaza administrarea pachetelor. Celelalte sunt simple comenzi specializate fiecare pe cate un aspect al gestionarii software. Evidenta pachetelor este tinuta in directorul */var/adm*, unde exista subdirectoarele *packages*, *scripts*, *removed_packages* si *removed_scripts*.

```
installpkg /pachete/openssh-3.5p1-i386-2.tgz
removepkg openssl                              (nu este necesar decat numele pachetului)
```

Pentru o mai mare usurinta in lucrul cu pachete, exista si in Slackware utilitare de tip integrator:

- **slackpgk** – permite instalarea/upgradarea pachetelor direct din retea, insa nu suporta dependinte
- **swaret** (Slackware Tool) – utilitar similar cu apt-get din Debian. Suporta si dependinte
- **slapt-get** – utilitarul apt-get portat pe Slackware

## 7.2.7. Integratoare de pachete

### *7.2.7.1. Concepte si avantaje*

Integratorul de pachete este un utilitar care automatizeaza procesul de instalare/dezinstalare/updgrade/etc. al pachetelor software prin detectarea dependintelor si instalarea/dezinstalarea automata a pachetelor necesare, acestea din urma putand fi descarcate automat in caz de nevoie de pe servere din internet (mirror-uri) definite de catre utilizator. Astfel, utilizatorul este degrevat de urmarirea interdependentelor si de travaliul obtinerii manuale a tuturor pachetelor necesare.

Pentru operatiile efective de instalare, dezinstalare etc, integratorul de pachete apeleaza tot la managerul de pachete, insa fata de acesta ofera servicii suplimentare:

- permite definirea mai multor surse de pachete. Acestea se pot afla intr-un sistem de fisiere local (partitii, CD, DVD, dispozitiv USB) sau pe un server din reteaua locala sau din internet, fiind accesibile prin HTTP, FTP, NFS etc
- detecteaza dependintele, descarcand si instaland automat pachetele necesare
- unele integratoare au capabilitatea de a detecta si dezinstala automat pachetele care nu mai sunt folosite (este in general cazul bibliotecilor de functii ce se instaleaza ca dependinte ale unui anumit pachet, dar dupa dezinstalarea pachetului raman in continuare in sistemul de fisere)

Sursele de pachete ale unui integrator sunt denumite “package repositories” (depozite de pachete) si reprezinta seturi de pachete aflate pe servere locale sau din internet (FTP, HTTP etc). Pachetele de pe server pot fi fie o copie a celor oficiale (caz in care spunem ca serverul este un *mirror* al celui oficial), fie un set de pachete suplimentar – spre exemplu, pachete puse la dispozitie de catre un anumit individ sau firma.

Odata dotat integratorul cu un set de repositories, acesta isi descarca listele de pachete disponibile si listele de fisiere cuprinse in acestea, alcatuind o baza de date locala, interogabila de catre utilizator. Asa se face ca, folosindu-ne de integrator, putem efectua operatii pe care simplul manager de pachete nu ni le poate oferi:

- cautare dupa nume sau fragment de nume sau de descriere in lista de pachete disponibile
- identificarea pachetului care contine un anumit fisier necesar

In distributiile Linux majore sunt intalnite diferite integratoare de pachete; ca minim, administratorul Linux al zilelor noastre trebuie  sa stapaneasca operatiile uzuale din cele doua mari familii de distributii - RedHat si Debian.

### *7.2.7.2. Familia RedHat*

In distributiile derivate din RedHat ne intalnim cu doua integratoare de pachete:

- **yum** (Yellowdog Updater Modified) - este un integrator de pachete folosit mult timp pe scara larga in distributiile RedHat, dar care este treptat inlocuit cu *dnf*
- **dnf** (Dandified Yum) - reprezinta o versiune imbunatatita a lui yum, introdusa odata cu Fedora 18 si devenit manager de pachete implicit incepand cu Fedora 22

Iata modul de realizare a principalelor operatii:

- instalare de pachet: **yum install numepachet** sau **dnf install numepachet**
- dezinstalare pachet: **yum remove numepachet** sau **dnf remove numepachet**

- cautare pachete dupa un fragment de nume sau de descriere: **yum search fragment** sau **dnf search fragment**
- afisare informatie despre un pachet: **yum info numepachet** sau **dnf info numepachet**
- listare dependinte pachet: **dnf repoquery --requires**
- sincronizare cu depozitele de pachete definite si actualizare baza de date pachete: **yum makecache** sau **dnf makecache**

### *7.2.7.3. Familia Debian*

APT contine o suita de utilitare pentru operatiile uzuale cu pachete:

- instalare pachet: **apt-get install *numepachet***
- dezinstalare pachet:
  - **apt-get remove *numepachet*** - sterge toate fisierele ce tin de pachet in afara de fisierele de configurare
  - **apt-get purge *numepachet*** – sterge inclusiv fisierele de configurare
- gasirea unui pachet pe baza unui fragment din nume:
  - **apt-cache search *fragment*** – afiseaza toate pachetele ce contin fragmentul specificat in numele sau in descrierea pachetului
  - **apt-cache search --names-only *fragment*** – cautarea se va efectua exclusiv in numele pachetelor
- afisarea informatiilor despre un pachet: **apt-cache show *numepachet***
- actualizare lista pachete: **apt-get update**

### *7.2.7.4. OpenSuse*

In OpenSuse, unealta folosita pentru administrarea pachetelor software este **zypper**, dupa cum urmeaza:

- instalare pachet: **zypper install numepachet**
- dezinstalare pachet: **zypper remove numepachet**
- cautare dupa fragment nume/descriere: **zypper search numepachet**
- afisare informatie pachet: **zypper info numepachet**
- actualizare lista pachete din repositories: **zypper refresh**

### *7.2.7.5. Concluzii*

Dupa cum se constata, exista diferente - pe alocuri considerabile - intre distributii cand vine vorba de managementul de soft, insa observam un fir rosu ce ne ajuta sa asimilam mult mai usor aceste diferente:

- toate sistemele prezentate presupun o lista de depozite de pachete, pe baza careia integratorul isi construieste o baza de date locala cu informatii despre totalul pachetelor disponibile
- operatiile dorite, cel putin cele uzuale, sunt aceleasi: instalare, dezinstalare, cautare, actualizare lista pachete
- cand vine vorba despre integratoare, sintaxa principalelor operatii difera in general prin numele utilitarului folosit: yum/dnf, zypper, apt/get/apt-cache. In rest, operatiile tind sa se numeasca install, remove si search

## *7.3. Compilarea si instalarea din surse*

## 7.3.1. De ce avem nevoie pentru a compila sursele

A putea compila soft pe o masina Linux presupune prezenta (cel putin a) urmatoarelor elemente:

- un compilator de C: in general, **gcc** (GNU C Compiler) si eventual **g++** (pentru C++)

- programe pentru generarea si analiza executabilelor, grupate in general in pachetul **binutils**
- **kernel headers:** o serie de fisiere in care sunt definite structuri si constante de sistem
- utilitarul **make** sau **cmake** – folosit pentru stabilirea dependentelor intre diversele fisiere ale unui proiect si compilarea acestora la nevoie

**make/cmake** este un utilitar care automatizeaza compilarea unui proiect. In lipsa lui ar trebui ca, pentru compilare, sa rulam comenzi brute de compilator. Acestea presupun in primul rand competente ce le depasesc pe cele uzuale ale unui administrator de sistem, si in plus pot atinge lungimi remarcabile, nefiind in consecinta practica rularea lor manuala.

Comanda *make* actioneaza conform unui fisier numit *Makefile*, care trebuie sa fie prezent in directorul rularii comenzii. Fisierul contine un set de tinte (targets), ce reprezinta operatii realizabile in directorul curent; fiecare target are asociata o lista de comenzi. Utilizatorul ruleaza comanda *make* urmata de un nume de target si, ca urmare, va fi rulat setul de comenzi corespunzator acelei tinte. Daca userul nu specifica un target, exista unul implicit, iar pentru surse de programe Linux acest target realizeaza compilarea intregului proiect (vezi 7.3.5).

Iata un exemplu de target:

```
psdocs: sgmldocs
	$(MAKE) -C Documentation/DocBook ps
```

Target-ul este *psdocs*; el depinde de un alt target, *sgmldocs* (specificat in acelasi Makefile). Un target poate fi un nume de fisier sau un alt nume ales de catre creatorul Makefile-ului. Linia a doua specifica comanda ce trebuie rulata in cazul acestui target; MAKE este o variabila definita anterior in fisier si care pointeaza catre utilitarul make. In cazul rularii comenzii **make psdocs**, make va verifica intai daca fisierele ce tin de target-ul sgmldocs s-au modificat, dupa care va rula comanda specificata in linia a doua.

## 7.3.2. Documentare prealabila

Inainte de a instala un soft din surse trebuie sa aflam cel putin aceste doua informatii:

- procedura de instalare - etapele de compilare si instalare, cat si optiunile disponibile pentru configurarea surselor si modalitatea lor de folosire. Desi majoritatea softurilor Linux tind sa respecte procedura de instalare prezentata in continuare, exista si destule exceptii
- lista de dependinte. Si softurile instalate din surse au o lista de cerinte care trebuie indeplinite pentru a se putea compila si instala corect

Informatiile pot fi aflate din cel putin doua surse:

- online, de pe site-ul softului sau al autorului sau. Este varianta recomandabila, deoarece informatia tinde sa fie mai bogata si intr-un format mai agreabil
- fisierele INSTALL si/sau README. In general, dupa dezarhivarea surselor, in directorul nou-aparut este prezent un fisier INSTALL sau/si unul README. Daca exista, fisierul INSTALL este cel care detaliaza procedura de compilare si instalare

## 7.3.3. Obtinerea si dezarhivarea surselor

Toate softurile open-source pun la dispozitia utilizatorului si sursele programului. Acestea pot fi gasite sub diferite forme:

- arhiva .tar.gz
- arhiva .tar.bz2
- pachet SRPM(source RPM): extensia .src.rpm

In primul si al doilea caz, dubla extensie este determinata de faptul ca sursele au fost mai intai arhivate cu utilitarul **tar** (care nu face decat sa le grupeze intr-un singur fisier, fara compresie) si apoi comprimate cu **gzip** sau **bzip2**. Obtinerea surselor presupune operatiunile corespunzatoare in ordine inversa:

1. Se decomprima sursele:

```
gunzip pachet.tar.gz                    ➔dispare extensia gz a fisierului
bunzip2 pachet.tar.bz2                  ➔dispare extensia bz2 a fisierului
```

2. Se dezarhiveaza fisierul tar:

```
tar -xpvf pachet.tar
```

In Linux, comanda **tar** suporta optiuni care realizeaza automat si decompresia:

```
tar -zxpvf pachet.tar.gz
tar -jxpvf pachet.tar.bz2
```

Semnificatiile optiunilor: **x** - extract, **v** – verbose (se afiseaza pe ecran detalii mai bogate despre operatia curenta), **f** – file (este urmat de numele fisierului), **p** - pastreaza permisiunile originale

Se obisnuieste ca sursele sa fie extrase in */usr/src* sau */usr/local/src* (directoare special create in acest sens).

## 7.3.4. Configurarea surselor in vederea compilarii

In functie de softul ce se doreste a fi instalat, aceasta etapa poate presupune:
- modificarea unor fisiere de configurare
- modificarea directa a fisierelor sursa
- folosirea scriptului **configure**, atunci cand softul in cauza foloseste GNU autoconf/automake (un set de utilitare create de GNU pentru usurinta scrierii de programe portabile). Aceasta este varianta folosita in majoritatea cazurilor

```
./configure [optiuni]
```

Scriptul *configure* primeste un set de argumente ce poate fi determinat ruland **./configure --help**. Iata cateva exemple:
- **--prefix** – stabileste locatia in care se va instala programul
- **--with-OPTIUNE**
- **--without-OPTIUNE** – permit activarea/dezactivarea unor facilitati ale softului. Sunt optiuni care nu mai pot fi schimbate odata ce softul a fost compilat.

Daca executia scriptului **configure** se incheie prematur cu o eroare, este recomandabila curatarea arborelui sursa inainte de a incerca o alta rulare a sa:

```
make distclean
```

***Nota:*** *chiar daca la rularea scriptului configure se stabileste directorul de instalare, crearea acestuia din urma se realizeaza abia in etapa de instalare efectiva!*

## 7.3.5. Compilarea

Compilarea este in general intermediata de comanda **make** sau **cmake**, rulata fara argumente. Ea verifica interdependentele fisierelor sursa si compileaza doar ceea ce este necesar. Vor rezulta, in arborele de fisiere sursa, fisiere .o (out) si executabilele finale.

```
make
```

Este posibila pornirea mai multor procese de compilare simultan – fapt util atunci cand statia dispune de mai multe procesoare sau de un procesor multi-core. Optiunea **–j** realizeaza acest lucru, iar parametrul ce-i urmeaza se situeaza uzual intre numarul de procesoare si dublul sau (solutia optima depinde de tipul procesoarelor prezente in sistem si de incarcarea sistemului):

```
make –j3     # pe o masina dual-core
```

Optional, unele softuri mai au un pas suplimentar, **make check**, care realizeaza verificarea functionalitatii programului inainte de a fi instalat.

Conform recomandarii de la punctul anterior, daca procesul de compilare se inrerupe cu erori, trebuie curatat arborele sursa inainte de o noua incercare.

***Nota:*** *fisierele Makefile sunt generate de catre scriptul configure (vezi mai jos), de aceea comanda make nu poate fi rulata imediat dupa dezarhivare.*

## 7.3.6. Instalarea

```
make install
```

Comanda copiaza la destinatia lor finala fisierele rezultate in urma compilarii, respectand optiunile specificate la pasul *./configure*.

Daca pasii anteriori pot fi efectuati ca user obisnuit (unele softuri chiar impun acest lucru ca masura de siguranta!), acest ultim pas in cele mai multe cazuri trebuie rulat ca root (pentru a avea drept de scriere in directoarele de sistem).

***Nota:*** *exista un program numit* ***checkinstall****, care urmareste instalarea unui soft din surse si poate crea automat pachete Debian, RPM sau tgz. Daca utilitarul checkinstall este disponibil, el poate fi folosit in locul comenzii make install. Efectul va fi crearea unui pachet si instalarea acestuia prin intermediul managerului de pachete, dezinstalarea devenind astfel facila.*

## 7.3.7. Curatarea arborelui surselor

In etapa de instalare, toate componentele softului au fost copiate din directorul sursa in directorul/directoarele destinatie, astfel ca in acest moment functionarea softului nu mai depinde de directorul surselor. In aceste conditii, dupa instalare se poate curata arborele surselor (fisierele intermediare *.o create de compilatorul C pot ocupa mult spatiu in cazul unui soft complex) sau chiar sterge sursele integral.

***Nota:*** *un motiv pentru care NU am dori sa stergem sursele este posibilitatea de dezinstalare a softului – vezi 7.3.9.*

In acest stadiu, utilizatorul are de obicei la dispozitie urmatoarele comenzi:

- **make clean** – sterge fisierele temporare create in urma compilarii, aducand directorul surselor la starea de imediat dupa rularea lui *configure*. Dispar fisierele .o si executabilele/bibliotecile rezultate, insa raman fisierele *Makefile* si configurarile efectuate prin intermediul lui configure (notabile fiind fisierele *config. cache* si *config.status*)
- **make distclean** – realizeaza o curatare in profunzime a arborelui surselor, aducandu-l in starea de dinainte de configure. In urma acestei operatii dispar inclusiv fisierele Makefile si cele cateva create de catre configure

## 7.3.8. Modificarea fisierelor de configurare si rularea programului

Majoritatea softurilor UNIX vin cu fisiere de configurare default, care insa trebuie cel putin copiate la locul lor sau/si modificate astfel incat sa corespunda cu dorintele noastre. Doar o parte dintre aplicatii pot porni „de la sine”, fara configurare anterioara.

## 7.3.9. Dezinstalare

Este posibil ca arborele surselor sa ofere posibilitatea de dezinstalare a softului. Daca in fisierul *Makefile* exista si target-ul *uninstall*, atunci utilizatorul poate folosi urmatoarea comanda pentru a automatiza dezinstalarea unui soft instalat din surse:

```
make uninstall
```

O alternativa este folosirea utilitarului *checkinstall* amintit mai sus.

## *7.4. BIBLIOGRAFIE*


- Tipuri si managere de pachete: http://www.linuxplanet.com/linuxplanet/tutorials/4161/1/
- RPM and DPKG command reference: http://packman.linux.is/
- Managere de pachete: http://en.wikipedia.org/wiki/Package_management_system
- Maximum RPM: http://www.rpm.org/max-rpm/
- Yum command cheat sheet: https://access.redhat.com/sites/default/files/attachments/rh_yum_cheatsheet_1214_jcs_print-1.pdf
- Zypper cheat sheet: https://en.opensuse.org/images/3/30/Zypper-cheat-sheet-2.pdf
- Package management cheatsheet: http://distrowatch.com/dwres.php?resource=package-management
- Tabel comparativ pentru diversele sisteme de management de soft: https://wiki.archlinux.org/index.php/Pacman/Rosetta

## 7.5. ANEXA 1 - comenzi pentru administrare soft in distributiile Linux uzuale

| Operatie | rpm | yum/dnf | dpkg | APT | zypper |
|---|---|---|---|---|---|
| Instalare pachet | rpm -ivh pachet.rpm | yum install pachet<br>dnf install pachet | dpkg -i pachet.deb | apt-get install pachet | zypper install pachet |
| Dezinstalare pachet | rpm -ev pachet | yum remove pachet<br>yum erase pachet | remove: dpkg -r pachet<br>purge: dpkg -P pachet | apt-get remove pachet<br>apt-get purge pachet | zypper remove pachet |
| Upgrade de pachet | rpm -Uvh pachet | yum update | | | |
| Cautare pachet neinstalat dupa fragment de nume/descriere | X | yum search fragment | X | apt-cache search fragment | zypper search fragment |
| De ce pachet apartine un fisier instalat | rpm -qf fisier | X | dpkg -S fisier | X | zypper what-provides fisier |
| Lista de fisiere dintr-un pachet | rpm -ql | X | dpkg -L numepachet | X | X |
| Lista de pachete instalate | rpm -qa | X | dpkg -l | X | X |
| Lista de dependinte ale unui pachet | rpm -R | repoquery --requires pachet | dpkg -I pachet.deb | apt-cache depends pachet | zypper info --requires pachet |
| Informatie despre un pachet | rpm -qpi | yum info pachet | dpkg -l | apt-cache show pachet | zypper info pachet |
| Fisiere configurare cu lista repositories | X | /etc/yum.repos.d | X | /etc/apt/sources.list | /etc/zypp/zypper.conf |
| Update liste din repositories | X | yum makecache | X | apt-get update | zypper refresh |
| Stergere pachete inutile | X | yum autoremove | X | apt-get autoremove | X |