**Aaron Andal**
**M05 – Fonaments de maquinari**

# Exercicis fitxers d'arxivament tar i compressió

Exercicis fitxers d'arxivament tar i compressió

Exercicis d'arxivament de fitxers (tar) i compressió/descompressió de fitxers (gzip, bzip2, xz)

**Introducció als fitxers d'arxivament i als fitxers de compressió**

Definició de format d'arxivament:

> És un format de fitxer que permet l'agrupament de fitxers en un de sol, però òbviament conservant la informació relativa al grup de fitxers (metadades).

S'utilitzen fitxers d'arxivament amb diferents finalitats com la creació de paquets de programari (*rpm*, *deb*), la creació de imatges de disc (*iso*) ...

A la wikipedia podem trobar una comparativa molt interessant

La següent taula ens mostra algunes característiques (*features*) dels diferents formats d'arxivament:

| Archive format | Built-in compression | Self-extracting | Directory Structure | POSIX attributes | ACLs | Alternate data streams |
|---|---|---|---|---|---|---|
| cpio | No[1] | No | Yes | Yes | No | ? |
| tar | No[1] | No | Yes | Yes | Some | (in Solaris implementation) |
| dar | Yes[3] | No | Yes | Yes | Yes | Yes |
| ar | No | No | No | Yes | No | ? |
| pax | No | No | Yes | Yes | Yes | ? |
| dump | No[1] | No | Yes | Yes | Yes | ? |
| shar | No | Yes | Yes | Yes | No | ? |
| makeself | Yes | Yes | Yes | Yes | Yes | ? |
| zip | Yes | Yes[2] | Yes | No | ? | ? |
| rar | Yes | Yes[2] | Yes | No | ? | Yes |
| ace | Yes | ? | Yes | No | ? | ? |
| arj | Yes | Yes[2] | Yes | No | No | ? |
| zoo | Yes | ? | Yes | No | ? | ? |
| ISO 9660 (CD-ROM) | No[1] | No | Yes | (with Rock Ridge extension) | No | ? |
| cab | Yes | Yes[2] | ? | No | ? | ? |
| rpm | Yes | No | Yes | Yes | ? | ? |
| deb | Yes | No | Yes | Yes | ? | ? |
| 7z | Yes | Yes | Yes | Yes | ? | ? |
| Archive format | Built-in compression | Self-extracting | Directory Structure | POSIX attributes | ACLs | Alternate data streams |

L'exemple per excel·lència i amb el que treballarem nosaltres serà el fitxer de tipus *tar* i per tant l'ordre `tar`.

Definició de format de compressió:

> Fitxer que conté la mateixa informació que l'original però utilitzant menys dades.

Hi ha molts formats de compressió, nosaltres treballarem amb *gzip*, *bzip2* i *xz*.

**Exercici -1 (el fem tots al mateix temps a classe)**

És molt probable que el nostre sistema de fitxers estigui format per blocs físics de mida 4KB. Ho podem comprovar amb diferents ordres:

```
fdisk -l /dev/sda
...
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
...
```

O també:

```
cat /sys/block/sda/queue/physical_block_size
4096
```

Anem a fer una prova, suposarem que no tenim cap altre servei al nostre sistema que pugui estar enviant fitxers a `/tmp`.

Fem un cop d'ull a la utilització de l'espai amb `df` (*disk free*):

```
df
... # massa informació
```

Filtrem, només mirem `/tmp` i anem creant fitxers i observant com evoluciona l'ús de l'espai de disc:

```
[john@i02 tmp]$  df | grep -w '/tmp'
tmpfs                                                          1969036     26272
1942764   2% /tmp
[john@i02 tmp]$ echo "soc un fitxer de pocs bytes que ocuparé 4K al disc dur" > f1
[john@i02 tmp]$  df | grep -w '/tmp'
tmpfs                                                          1969036     26276
1942760   2% /tmp
[john@i02 tmp]$ echo "soc un fitxer de pocs bytes que m'agrada tenir molt d'espai
lliure al meu sector" > f2
[john@i02 tmp]$  df | grep -w '/tmp'
tmpfs                                                          1969036     26280
1942756   2% /tmp
[john@i02 tmp]$ echo "Soc un fitxer sobradíssim" > f3
[john@i02 tmp]$  df | grep -w '/tmp'
tmpfs                                                          1969036     26284
1942752   2% /tmp
[john@i02 tmp]$ echo -e "soc un fitxer de pocs bytes que ocuparé 4K al disc
dur\nsoc un fitxer de pocs bytes que m'agrada tenir molt d'espai lliure al meu
sector\nSoc un fitxer sobradíssim\n" > f123
[john@i02 tmp]$  df | grep -w '/tmp'
```

```
tmpfs                                                     1969036     26288
1942748   2% /tmp
```

Calculem la mida dels fitxers que acabem de crear (també ho podríem fer amb `wc -c`):

```
[john@i02 tmp]$ ls -l /tmp/f[1-3]*
-rw-r--r--. 1 john inf  56 Feb  5 15:30 /tmp/f1
-rw-r--r--. 1 john inf 165 Feb  5 15:32 /tmp/f123
-rw-r--r--. 1 john inf  81 Feb  5 15:30 /tmp/f2
-rw-r--r--. 1 john inf  27 Feb  5 15:31 /tmp/f3
```

Veiem que 3 fitxers de 56, 81 i 27 bytes ocupen 4K + 4K + 4K en disc, o sigui 12K, i en canvi un únic fitxer de la mateixa mida (de fet un byte més) *només* ocupa un bloc de 4K.

Així es pot concloure que aconseguir transformar un grup de fitxers en únic de sol ja té al menys un avantatge: **estalvi d'espai al disc dur**.

Recordem que el sistema de fitxers divideix la nostra partició en blocs i que donat un fitxer s'utilitza almenys un bloc. Aquest bloc és exclusiu d'aquest fitxer. Si sobra espai i en un futur es modifica el fitxer s'utilitza aquest bloc i si s'omple es reserva un altre bloc per a aquest fitxer. Tot i que pugui semblar un malbaratament d'espai, aquest disseny té molts avantatges, per exemple al tenir tots els blocs (registres) la mateixa mida el seu accés sempre és directe (no seqüencial) i per tant és més ràpid.

**Exercici 0**

Des de *Fedora* i *Debian* comproveu si teniu instal·lat al vostre sistema la documentació de *postgresql*.

El paquet a *Fedora 24* es diu `postgresql-docs` i a *Debian Stretch* `postgresql-doc`.

Comproveu amb l'ordre adient si està instal·lat el paquet o no. Si no està instal·lat, baixeu-lo utilitzant el gestor de paquets del sistema en el qual us trobeu **sense instal·lar res**.

No el volem instal·lar. Volem obrir i descomprimir el paquet. Ho farem gràficament amb `file-roller` descomprimint a l'Escriptori. Podeu fer un cop d'ull amb el navegador per veure aquesta ajuda (heu de cercar on es troba el fitxer `index.html`.

```
[root@j20 ~]# rpm -qa | grep postgresql
postgresql-9.5.7-1.fc24.x86_64
postgresql-libs-9.5.7-1.fc24.x86_64
postgresql-server-9.5.7-1.fc24.x86_64
[root@j20 ~]#
Està instal·lat.

Per tant pel primer cercant una eina per descomprimir trobem `rpm2cpio`:
rpm2cpio postgresql-docs-XXXXXXXXXXXX.rpm | cpio -dium

Per descomprimir el paquet `deb` podem fer servir l'ordre `ar`:
ar postgresql-doc-XXXXXXXXXXXX.deb
```

**Exercici 1**

L'exercici anterior haurà creat un directori `usr` amb diferents subdirectori si fitxers. Quina ordre em crea un fitxer de tipus *tar* (sense comprimir)? (Hint: `man tar`)

tar --create [--file ARCHIVE] [OPTIONS] [FILE...]
locate postgresql | grep postgresql

/usr/libexec/initscripts/legacy-actions/postgresql/upgrade
/usr/share/postgresql-setup
/usr/share/augeas/lenses/dist/postgresql.aug
/usr/share/doc/postgresql
/usr/share/doc/postgresql-libs
/usr/share/doc/postgresql/COPYRIGHT
/usr/share/doc/postgresql/HISTORY
...

**Exercici 2**

Quina ordre em fa el mateix d'abans però comprimint amb gzip? I amb bzip2? I amb xz? Després de comprimir amb les 3 ordres fixa't bé en el procés de compressió de tots 3. Hi ha alguna diferència de velocitat en l'execució de l'ordre (recorda que si poses `time` davant d'una ordre et mostra el temps que triga l'ordre)? I en la mida, hi ha diferències? Pots concloure que un compressor serà millor que altre sempre? o depén de l'ús que es vulgui fer? (Hint: `man tar`)

Gzip -c --stdout
bzip2 -z –compress
xz -z –compress

Es fa amb la ordre time abans de les 3 comandes

**Exercici 3**

Continuem treballant a l'escriptori. Si tenim aquí el directori `usr` d'abans eliminem-lo.

- Descomprimiu el fitxer d'extensió *tar.gz* al mateix escriptori.

Amb la ordre Tar -xzvf usr.tar.gz

- Feu el mateix amb el fitxer d'extensió *xz* però sense canviar el directori actiu (escriptori) volem que es descomprimeixi a `/tmp`.

Amb la ordre xz -z usr.xz –durectory /tmp

**Exercici 4**

Quina ordre ens permet llistar el contingut d'un tar sense necessitat de descomprimir-lo?

La ordre qu ens permet llistar el contingut d'un tar sense necessitat de descomprimir-ho es amb l'ordre tar –list -f

**Exercici 5**

Tenim estructures de directoris transformades en tar's comprimits. Quines ordres aconseguiran només descomprimir els diferents comprimits que tinc? I un cop tingui un tar, com puc passar del tar a l'estructura de directoris original?

Per descomprimir --> Tar -xvf --> Per poder transformar un tar a l'estructura de directoris original hem de fer --> tar -xzvf

**Exercici 6**

Executeu les següents ordres:

```
[john@i02 Desktop]$ cd ~/Desktop/
[john@i02 Desktop]$ mkdir -p  exercici6/dir1/dir11/dir12/{dir121,dir122}
[john@i02 Desktop]$ cd exercici6/
[john@i02 exercici6]$ echo "soc el fitxer fitxer1" >
dir1/dir11/dir12/dir121/fitxer1.txt
[john@i02 exercici6]$ tree
.
└── dir1
    └── dir11
        └── dir12
            ├── dir121
            │   └── fitxer1.txt
            └── dir122

5 directories, 1 file
```

- Descriu que fa cadascuna d'aquestes ordres.
- Sense moure'ns de directori, quina ordre crearà un fitxer *tgz* que contigui tota l'estructura de directoris i fitxers (sense el directori *exercici6*)

Cd $HOME/Desktop/ --> Es mou a l'escriptori
mkdir -p  exercici6/dir1/dir11/dir12/{dir121,dir122} -->
Crear directoris diferents entre els {} dins d'un directori – L'opció -p (parents), té la funció de detectar que no hi hagi erros.
Cd exercici6/ --> Canvia de directori
`echo "soc el fitxer fitxer1" > dir1/dir11/dir12/dir121/fitxer1.txt --> Afegeix la cadena de caràcters "soc el fitxer fitxer1" a l'arxiu`

```
dir1/dir11/dir12/dir121/fitxer1.txt
tree --> Visualitzar l'estructura de fitxers.
```

**Exercici 7**

Es pot indicar el nivell de compressió o de velocitat quan s'utilitzen *gzip*, *bzip2* i *xz*? I per descomprimir?

Amb l'ordre gzip, l'opció -# es por regular la velocitat de compressió utilitzant el dígit especificat

Si utilitzem l'ordre xz, en les opcions de velocitat tenim que aplicarles amb un interval entre el -0 -9 exemple: xz -5 (arxiu.xz)

**Exercici 8**

Baixa el paquet r.tar.gz i descomprimeix tot el que es pugui descomprimir.

```
tar xzf r.tar.gz r/r.tar.gz r/r.tar.gz r/r.tar.gz
```