# TAR

Descompressor d’arxius tar

Daniel Carrillo Cardús
CFGS Desenvolupament Web | Es Liceu
2019-2020

# Introducció

En informàtica, el format de fitxers tar, és un dels formats per arxivament, el nom prové de Tape ARchive format (format d'enregistrament de cintes). Aquests arxius es produeixen amb la comanda Unix tar i es van estandarditzar en l'especificació POSIX.1-1998 i més endavant per POSIX.1-2001. Es fa servir àmpliament per empaquetar i desempaquetar fitxers, per tal d'acumular un llarg nombre de fitxers dins d'un sol fitxer tot preservant l'estructura de fitxers, els permisos, les propietats i les dates. Seguint la Filosofia Unix, consistent en saber "fer una sola cosa" (empaquetar) però "fer-la bé", no incorpora compressió de manera que cal combinar-la amb altres eines, per aconseguir un empaquetament comprimit.

El que farem serà un programa que fagi feina amb arxius Tar i interactui amb ells, per descomprimir-los, per carregar-los a memòria, i més.

L'enllaç a GitHub d'aquest projecte el podeu trobar al següent enllaç.

github.com/dcarrilloc/wintar

# Classe Tar

Aquesta classe tindrà tres mètodes principals per poder fer feina amb tars. El primer és extract().

## Expand()

Aquest mètode és el mètode més important, ja que s'encarrega de carregar el tar amb el que farem feina a memòria. Per fer això crearem una llista de Documents. Un Document és un objecte que representa un arxiu dins d'un Tar, per exemple, una imatge, un vídeo, una cançó, un fitxer de text, etc...

Els arxius dins un tar estan representats en forma de bytes. Per poder diferenciar quan comença i quan acaba un fitxer o el següent hem de seguir aquesta taula:

| Field offset | Field size | Field |
|---|---|---|
| 0 | 100 | File name |
| 100 | 8 | File mode |
| 108 | 8 | Owner's numeric user ID |
| 116 | 8 | Group's numeric user ID |
| 124 | 12 | File size in bytes (octal base) |
| 136 | 12 | Last modification time in numeric Unix time format (octal) |
| 148 | 8 | Checksum for header record |
| 156 | 1 | Link indicator (file type) |
| 157 | 100 | Name of linked file |

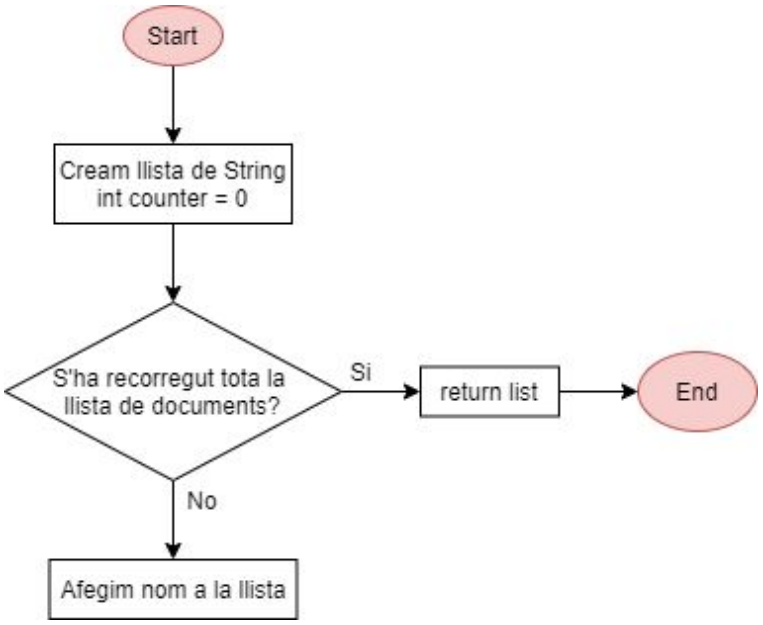
Sabent com estan distribuïts els bytes que ens trobarem podem saber on comença i on acaba cada arxiu.

```
Agafam nom en els següents 100 bytes;
if(nom = "") {
    Comprovam que els següents 412 bytes estan buits
    if(seguents 412 bytes = "") {
        Final del Tar;
    }
}
Agafam filemode en els següents 8 bytes;
Agafam propietari en els següents 8 bytes;
Agafam grup en els següents 8 bytes;
Agafam tamany en els següents 12 bytes;
Agafam darrera modificacio en els següents 12 bytes;
Agafam checksum en els següents 8 bytes;
Agafam link indicator en el següent 1 byte;
Agafam nom de arxiu enllaçat en els següents 100 bytes;
Botam bytes necessaris per sortir del header (255);
Agafam content;
Cream objecte amb tota la informació anterior;
Botam bytes necessari per començar el següent arxiu;
```

## List()

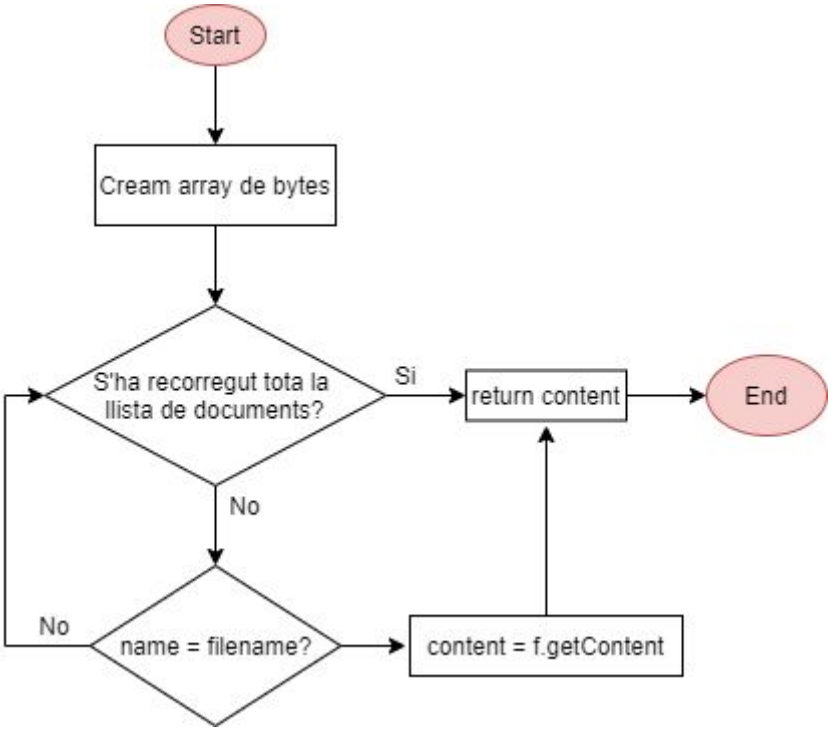
Aquest mètode serveix per llistar els arxius que es troben dins un fitxer. Només caldrà recórrer la llista de Documents (que és un atribut de la classe Tar) i anirem agafant l'atribut "*filename*" de cada Document.

En aquest cas, degut a la bona recopilació de dades al mètode anterior "extract()", és molt fàcil obtenir la informació que cercam. Seria de la següent manera:




## GetBytes()

Un cas molt similar es el de getBytes(). Aquest mètode retorna un array de bytes amb el contingut del fitxer que té per nom igual a l'String «name» que passem per paràmetre.



Fins aquí hem construït la clase que fa feina amb els arxius Tar. Ara anem a posar a prova aquest codi creant un programa interactiu que se relacioni amb aquesta clase.

# Programa principal

Aquest programa principal es compondrà d'una sèrie de mètodes segons la comanda que l'usuari demani per consola (aquest programa no té interfaz gràfica). Anem a veure el menú principal.

```
Benvolgut. Quina operació vols fer?
=====================================================================
1-    Carregar un fitxer tar a memòria. (OBLIGATORI EL PRIMER PIC)
2-    Llistar els nombres dels arxius que hi ha dins el fitxer.
3-    Conéixer el tamany dels arxius.
4-    Conéixer el propietari dels arxius.
5-    Conéixer el grup dels arxius.
6-    Extreure el fitxer a una destinació.
7-    Obrir un arxiu contingut al tar.
8-    Edita un arxiu contingut al tar.
9-    Imprimeix un arxiu contingut al tar. (No es vàlid per a tots els arxius).
10-   Sortir.
=====================================================================
```

Aquestes son totes les comandes que pots realitzar. Cal destacar que el menú està dintre d'un bucle infinit per a que es pugui realitzar varies comandes sense haver d'aturar el programa. Es sortirà d'es programa quan es mandi la comanda 10.

Cada comanda està dirigida per un bloc de decisió *switch* que segons la comanda enviarà les instruccions al compilador d'executar un mètode o un altra.

Cal destacar que per que funcioni el programa es obligatori primer executar la comanda 1, que es la que s'encarrega de carregar el arxiu Tar a memòria per poder fer feina amb ell més endavant.

## load()

En aquest mètode bàsicament li demanam a l'usuari que introdueixi la ruta de l'arxiu Tar que vol carregar i cridam al mètode *extract()* de la classe Tar que hem creat anteriorment.

## list()

Aquest mètode llista els fitxers que hi ha dins el Tar recorrent la llista de fitxers que té el nostre objecte Tar i imprimint per pantalla el nom de cada fitxer.

## size()

Llista els tamanys dels fitxers que hi ha dins el tar. Per això recorrem la llista de fitxers que té el nostre objecte Tar imprimint per pantalla el tamany de cada fitxer.

## owner()

Llista els propietaris dels fitxers que hi ha dins el tar. Per fer això recorre la llista de fitxers del nostre objecte Tar i agafa l'atribut *fileOwner* de cadascun.

## group()

El mateix passa a l'hora de saber el grup de cada fitxer. Recorrerem la llista i agafarem l'atribut *fileGroup* de cada arxiu.

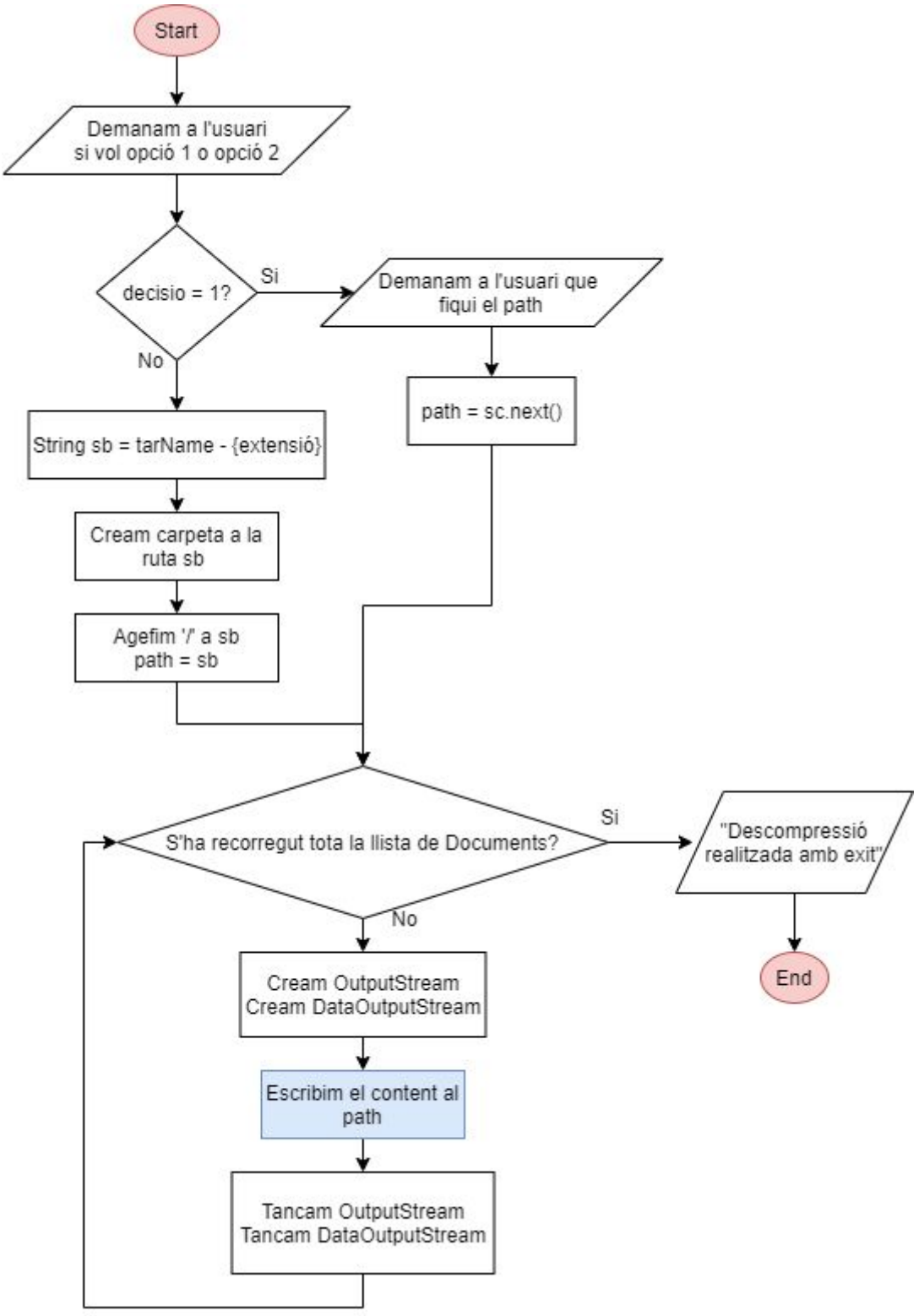
## extract()

El mètode extract ja és més interessant. Serveix per extreure un fitxer del Tar a una destinació que li passem. En aquest mètode podem trobar un try-catch ja que algunes classes emprades per OutputStream poden generar excepcions.

Primer demanam a l'usuari la manera en que vol descomprimir el Tar. Hi ha dues opcions: a la mateixa ruta del Tar o a una altra diferent (a gust de l'usuari).

Depenent de l'opció que agafi l'usuari crearem un *path*, que només és la destinació on descomprimirem el Tar. Si l'usuari demana descomprimir l'arxiu en la mateixa ubicació que l'arxiu Tar (homònim de l'eina "*Extraer aquí*" del famós programa WinRar), el path agafarà mitjançant StringBuilder el path de l'arxiu Tar i crearà una carpeta amb el mateix nom on ficar els arxius descompressos. Si l'usuari demana la segona opció serà ell qui ficarà manualment el path on vol desar els fitxers i després es crearà la nova carpeta on ficar els arxius.

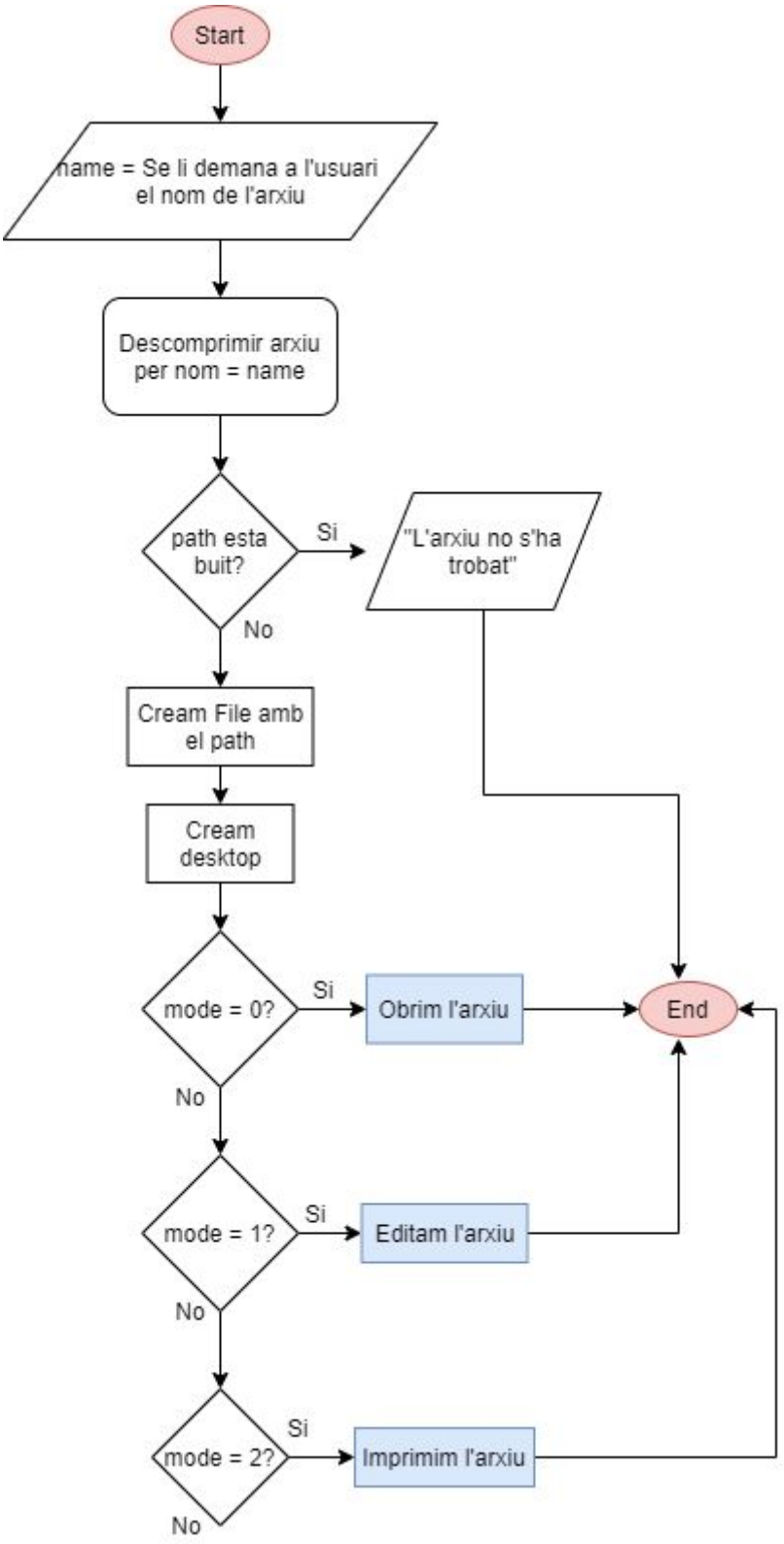
La següent passa és recórrer la llista de Documents dins el Tar i anar descomprimint-los al path personalitzat (acaba amb el nom de l'arxiu). Per fer això emprarem OutputStream i DataOutputStream, escriurem el contingut de cada fitxer al path, i tancarem els OutputStream i DataOutputStream. Aquesta passa és clau, ja que sense tancar-los no es podrien veure bé els arxius fins que s'aturi el programa.


Start
Demanam a l'usuari
si vol opció 1 o opció 2
decisio = 1?
Si
No
Demanam a l'usuari que
fiqui el path
path = sc.next()
String sb = tarName - {extensió}
Cream carpeta a la
ruta sb
Agefim '/' a sb
path = sb
S'ha recorregut tota la llista de Documents?
Si
No
"Descompressió
realitzada amb exit"
End
Cream OutputStream
Cream DataOutputStream
Escribim el content al
path
Tancam OutputStream
Tancam DataOutputStream

## run()

Aquest mètode fa tres operacions principals: obrir, editar, o imprimir un arxiu que es troba dins un Tar, i respón a les comandes 7, 8, i 9 del menú. Degut a les similituds entre les tres operacions s'ha obtat per emprar el mateix mètode passant-li per paràmetre quina de les tres operacions ha de realitzar. Si el paràmetre és 0 s'edita l'arxiu, si és 1 s'obri, i finalment, si és 2 s'imprimeix. També cal destacar que abans de poder realitzar qualsevol de les tres operacions citades anteriorment cal descomprimir primer l'arxiu.

Per realitzar l'operació desitjada per l'usuari podem emprar la classe Desktop, que bàsicament s'encarrega de mantenir una "relació" amb el sistema operatiu per executar unes ordres o unes altres, com poden ser obrir, editar, imprimir, etc...


Start
name = Se li demana a l'usuari el nom de l'arxiu
Descomprimir arxiu per nom = name
path esta buit?
Si
"L'arxiu no s'ha trobat"
No
Cream File amb el path
Cream desktop
mode = 0?
Si
Obrim l'arxiu
End
No
mode = 1?
Si
Editam l'arxiu
No
mode = 2?
Si
Imprimim l'arxiu
No

# Conclusions

Aquest projecte m'ha resultat molt divertit ja que, al contrari de la majoria de pràctiques fetes fins ara, aquesta pràctica té un plaer visual, ja que tu veus com es descomprimeix, com es creen carpetes, etc...

També ha resultat molt profitosa en l'àmbit d'entrada i sortida de dades amb Java i l'ús adequat de les excepcions.