

TABLE DES MATIÈRES

TABLE DES MATIÈRES.....	1
Introduction à la compression de données.....	2
1. Définition.....	2
2. Quel est l'utilité de la compression des fichiers.....	2
3. Comment fonctionne la compression des fichiers.....	3
3.1. La compression sans perte.....	3
3.2. La compression avec perte.....	4
Aperçu des outils de compression de fichiers courants sous Linux.....	6
1. GZIP.....	6
1.1. PRÉSENTATION.....	6
1.2. COMPRESSION ET DÉCOMPRESSION AVEC GZIP.....	6
1.2.1. COMPRESSION.....	6
1.2.1.1. Compression dans l'arborescence de répertoires.....	8
1.2.1.2. Utilisation avec tar.....	9
1.2.2. DÉCOMPRESSION.....	10
1.2.2.1. Décompression et écrasement.....	11
2. BZIP2.....	11
2.1. PRÉSENTATION.....	11
2.2. COMPRESSION ET DÉCOMPRESSION AVEC BZIP2.....	12
2.2.1. COMPRESSION.....	12
2.2.2. DÉCOMPRESSION.....	12
3. GNU TAR.....	13
3.1. PRÉSENTATION.....	13
3.2. CARACTÉRISTIQUES TECHNIQUE.....	13
3.3. COMPRESSER ET DÉCOMPRESSER LES FICHIERS DANS UNE ARCHIVE TAR.....	14
3.3.1. Gérer les archives .tar avec un logiciel graphique.....	14
3.3.2. Utilisation en ligne de commandes.....	14
3.3.2.2. tar : extraction de fichiers.....	15
3.3.2.3. Compression avec gzip (.tar.gz).....	15
3.3.2.4. Compression avec Bzip2 (.tar.bz2).....	15
3.3.2.5. Compression avec Lzma (.tar.xz).....	15

Introduction à la compression de données

1. Définition

La compression de données est un processus informatique qui transforme une suite de bits en une suite plus courte pouvant restituer les mêmes informations grâce à un algorithme de décompression. Cela réduit la taille des données pour la transmission ou le stockage. La décompression est l'opération inverse.

2. Quel est l'utilité de la compression des fichiers

Si vous êtes une personne qui édite fréquemment les fichiers, ou les transfère fréquemment, vous n'êtes pas sans savoir que la compression des fichiers est cruciale pour le gain tant du temps d'envoi que de l'espace de stockage. N'ayez crainte si vous ne faites pas partie de ces personnes, nous allons voir dans les lignes qui suivent pourquoi c'est si nécessaire:

- **Occupation de moins d'espace:** Ouais, vous avez bien lu. Imaginez que vous avez une tonne de fichiers que vous prévoyez de ne pas utiliser après un certain bout de temps, peut-être vos STAR WARS, STAR TREK et SEIGNEURS DES ANNEAUX et aussi peut-être toutes les saisons de DOCTOR WHO. Les laisser sur le disque dur pour occuper votre disque dur ne serait pas le choix le plus pratique à faire. Par contre, vous pouvez compresser, ou ZIPPER pour les habitués, un grand nombre de fichiers dans un seul dossier archive, ce qui à la fois permet une meilleure organisation.
- **Transferts plus efficaces:** Si vous avez essayé d'envoyer des fichiers grâce à une clé usb, vous n'êtes pas sans savoir que c'est une opération qui peut prendre beaucoup de temps. Pire encore, si le nombre de fichiers que vous essayez d'envoyer est conséquent, vous verrez le temps de transfert augmenter. Certes, vous pouvez envoyer les fichiers petit à petit mais cela est difficile à suivre. Mais en compressant le fichier dans un archive il prendra moins d'espace et le transfert sera beaucoup plus rapide. Et vous n'aurez qu'à décompresser le fichier dans l'appareil destinataire.

Une autre de ces utilités pourrait être de faire des économies mais cela dépend de ce que nous faisons de la compression des données.

3. Comment fonctionne la compression des fichiers

C'est là que les choses peuvent se compliquer un peu. Il existe essentiellement deux grands types de compression de fichiers comme mentionné ci-haut: la compression sans perte, et la

compression avec perte. La compression sans perte prend vos fichiers et réduit leur taille sans perdre aucune information. La compression avec perte réduit la taille de votre fichier en coupant les bits et les morceaux qui ne sont pas nécessaires à 100% pour fonctionner. Je sais que c'est un peu trop simplifié, alors analysons les choses une par une.

Un algorithme de compression sans perte permet de récupérer exactement les données d'origine après décompression. Ces algorithmes sont utilisés pour les archives, les fichiers exécutables et les textes. Avec un algorithme de compression avec perte, les données obtenues après décompression sont proches de l'original selon la qualité désirée. Ces algorithmes sont utiles pour les images, le son et la vidéo. Des formats tels que Zip, RAR, gzip, ADPCM, MP3 et JPEG utilisent des algorithmes de compression de données.

3.1. La compression sans perte

La compression est dite sans perte lorsqu'il n'y a aucune perte de données sur l'information d'origine. Il y a autant d'information après la compression qu'avant, elle est seulement réécrite d'une manière plus concise (c'est par exemple le cas de la compression gzip pour n'importe quel type de données ou du format PNG pour des images synthétiques destinées au Web2). La compression sans perte est dite aussi compactage.

L'information à compresser est vue comme la sortie d'une source de symboles qui produit des textes finis selon certaines règles. Le but est de réduire la taille moyenne des textes obtenus après la compression tout en ayant la possibilité de retrouver exactement le message d'origine (on trouve aussi la dénomination codage de source en opposition au codage de canal qui désigne le codage correcteur d'erreurs). Un algorithme de compression sans perte pourrait permettre de prendre un fichier écrit comme ceci :

```
AAAAABBBJJEEE
```

Et le compresser comme cela:

```
A5B3J2E3
```

Cependant, il n'existe pas de technique de compression de données sans perte universelle, qui pourrait compresser n'importe quel fichier : si une technique sans perte compresse au moins un fichier, alors elle en « grossit » également au moins un autre.

3.2. La compression avec perte

La compression avec pertes ne s'applique qu'aux données « perceptibles », en général sonores ou visuelles, qui peuvent subir une modification, parfois importante, sans que cela soit perceptible par un humain. La perte d'information est irréversible, il est impossible de retrouver les données d'origine après une telle compression. La compression avec perte est pour cela parfois appelée compression irréversible ou non conservative.

Cette technique est fondée sur une idée simple : seul un sous-ensemble très faible de toutes les images possibles (à savoir celles que l'on obtiendrait par exemple en tirant les valeurs de chaque pixel par un générateur aléatoire) possède un caractère exploitable et informatif pour l'œil. Ce sont donc ces images-là qu'on va s'attacher à coder de façon courte. Dans la pratique, l'œil a besoin

pour identifier des zones qu'il existe des corrélations entre pixels voisins, c'est-à-dire qu'il existe des zones contiguës de couleurs voisines. Les programmes de compression s'attachent à découvrir ces zones et à les coder de la façon aussi compacte que possible. La norme **JPEG 2000**, par exemple, arrive généralement à coder des images photographiques sur 1 bit par pixel sans perte visible de qualité sur un écran, soit une compression d'un facteur 24 à 1.

Puisque l'œil ne perçoit pas nécessairement tous les détails d'une image, il est possible de réduire la quantité de données de telle sorte que le résultat soit très ressemblant à l'original, voire identique, pour l'œil humain. L'enjeu de la compression avec pertes est de réduire la quantité de données d'un fichier tout en préservant la qualité perceptible et en évitant l'apparition d'artefacts.

De même, seul un sous-ensemble très faible de sons possibles est exploitable par l'oreille, qui a besoin de régularités engendrant elles-mêmes une redondance (coder avec fidélité un bruit de souffle n'aurait pas grand intérêt). Un codage éliminant cette redondance et la restituant à l'arrivée reste donc acceptable, même si le son restitué n'est pas en tout point identique au son d'origine.

On peut distinguer trois grandes familles de compression avec perte :

- par prédiction, par exemple l'ADPCM (Adaptive differential code modulation);
- par transformation. Ce sont les méthodes les plus efficaces et les plus utilisées. (**JPEG, JPEG 2000**, l'ensemble des normes **MPEG...**) ;
- compression basée sur la récurrence fractale de motifs (Compression fractale).

Les formats MPEG sont des formats de compression avec pertes pour les séquences vidéos. Ils incluent à ce titre des codeurs audio, comme les célèbres MP3 ou AAC, qui peuvent parfaitement être utilisés indépendamment, et bien sûr des codeurs vidéos — généralement simplement référencés par la norme dont ils dépendent (MPEG-2, MPEG-4), ainsi que des solutions pour la synchronisation des flux audio et vidéo, et pour leur transport sur différents types de réseaux.

Aperçu des **outils** de **compression** de **fichiers** courants sous **Linux**

Sous Linux, existe plusieurs outils de compression de fichiers tous plus performants les uns que les autres. Mais tout au long de ce travail nous n'en verrons que trois qui sont les plus connus en l'occurrence gzip, bzip2 et le dernier mais pas des moindres, tar.

1. GZIP

1.1. PRÉSENTATION

gzip (acronyme de GNU zip) est un logiciel libre de compression qui a été créé à partir de 1991 pour remplacer le programme compression d'Unix. **Gzip** est basé sur l'algorithme deflate, qui est une combinaison des algorithmes **LZ77** et **Huffman**. 'Deflate' a été développé en réponse à des problèmes de brevet logiciel couvrant **LZW** et autres algorithmes de compression, limitant ainsi les utilisations possibles de compression et autres programmes d'archivage populaires.

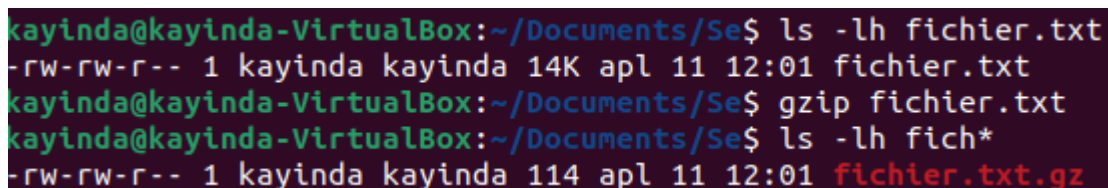
De manière à simplifier les développements de logiciels utilisant la compression, la bibliothèque zlib a été créée. Elle supporte le format de fichier gzip et l'algorithme de compression deflate. Cette bibliothèque est très largement utilisée, grâce à sa taille réduite, son efficacité et sa souplesse d'utilisation. gzip comme zlib ont été écrits par Jean-Loup Gailly et Mark Adler.

1.2. COMPRESSION ET DÉCOMPRESSION AVEC GZIP

1.2.1. COMPRESSION

Pour compresser un fichier il suffit de passer le nom du fichier à la commande. Nous allons aussi vérifier la taille d'origine du fichier, le compresser, puis vérifier la taille du fichier .gzip. Les commandes que nous allons utiliser sont:

```
ls -lh fichier.txt
gzip fichier.txt
ls -lh fich*
```



```
kayinda@kayinda-VirtualBox:~/Documents/Se$ ls -lh fichier.txt
-rw-rw-r-- 1 kayinda kayinda 14K apl 11 12:01 fichier.txt
kayinda@kayinda-VirtualBox:~/Documents/Se$ gzip fichier.txt
kayinda@kayinda-VirtualBox:~/Documents/Se$ ls -lh fich*
-rw-rw-r-- 1 kayinda kayinda 114 apl 11 12:01 fichier.txt.gz
```

Le fichier d'origine, une feuille de calcul appelée « calc-sheet.ods », est de 11 Ko et le fichier compressé, également appelé fichier d'archive, est de 9,3 Ko. Notez que le nom du fichier d'archive est le nom du fichier d'origine auquel est ajouté « .gz ».

La première utilisation de la commande cible un fichier spécifique, la feuille de calcul. La deuxième utilisation de recherche tous les fichiers commençant par « calc- » mais ne trouve que le fichier compressé. En effet, par défaut, crée le fichier d'archive et supprime le fichier d'origine.

Si vous avez besoin du fichier d'origine, vous pouvez le récupérer à partir du fichier d'archive. Mais si vous préférez conserver le fichier d'origine, vous pouvez utiliser l'option -k.

Les commandes que nous avons utilisé sont:

```
$ gzip -k fichier.txt
$ ls -lh fich*
```

```
kayinda@kayinda-VirtualBox:~/Documents/Se$ gzip -k fichier.txt && ls -lh fich*
-rw-rw-r-- 1 kayinda kayinda 14K apl 11 12:10 fichier.txt
-rw-rw-r-- 1 kayinda kayinda 114 apl 11 12:10 fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se$
```

Cette fois le fichier d'origine est conservé.

Pours zipper un fichier.txt avec un taux de compression maximum, nous pouvons utiliser l'argument -9:

```
$ gzip -9 fichier.txt
```

Remarque: Plus vous augmentez le taux de compression, moins le temps de compression est réduit et inversement. c'est à dire que si nous essayons de mettre taux de compression à -1 par exemple, cela prendra moins de temps mais par contre mon fichier .gz ne sera pas compressé à son minimum.

C'est un peu comme quand vous rangez des habits dans une valise, les habits représentent les bits et la valise la taille initiale du fichier du fichier. Le taux de compression est la façon dont vous pliez les habits et -1 signifie peut-être habits non pliés du tout. Vous savez peut-être mieux que moi que ces habits seront rangés dans la valise mais nécessitera beaucoup plus d'espaces dans la valise que si vous les aviez pliés.

1.2.1.1. Compression dans l'arborescence de répertoires

L'option (récursive) entraîne la compression des fichiers dans une arborescence de répertoires entière. Mais le résultat pourrait ne pas être ce à quoi vous vous attendiez.-r gzip

Voici l'arborescence de répertoires que nous allons utiliser dans cet exemple. Les répertoires contiennent chacun un fichier texte.

```
$ tree level1
```

```
kayinda@kayinda-VirtualBox:~/Documents/Se2$ tree level2
level2
├── depth1.txt
├── level2-1
│   ├── depth2-1.txt
│   │   └── level3-1-1
│   │       └── depth3-1-1.txt
│   └── level2-2
│       ├── depth2-2.txt
│       ├── level3-2-1
│       │   └── depth3-2-1.txt
│       └── level3-2-2
│           └── depth3-2-2.txt
5 directories, 6 files
```

Utilisons sur cet arborescence des répertoires et voyons ce qui se passe avec gzip

```
$ gzip -r level1/
```

```
$ tree level1
```

```
kayinda@kayinda-VirtualBox:~/Documents/Se2$ gzip -r level1/
kayinda@kayinda-VirtualBox:~/Documents/Se2$ tree level1
level1
├── depth1.txt.gz
├── level2-1
│   ├── depth2-1.txt.gz
│   │   └── level3-1-1
│   │       └── depth3-1-1.txt.gz
│   └── level2-2
│       ├── depth2-2.txt.gz
│       ├── level3-2-1
│       │   └── depth3-2-1.txt.gz
│       └── level3-2-2
│           └── depth3-2-2.txt.gz
5 directories, 6 files
```

Le résultat est a créé un fichier d'archive pour chaque fichier texte dans la structure de répertoires. Il n'a pas créé d'archive de l'arborescence de répertoires entiers. En fait, on ne peut mettre qu'un seul fichier dans une archive. Dans la section suivante nous verrons comment faire pour compresser un dossier ayant une arborescence de fichiers en un seul fichier d'archive compressé.

1.2.1.2. Utilisation avec tar

Nous pouvons créer un fichier d'archive qui contient une arborescence de répertoires et tous ses fichiers, mais nous devons mettre une autre commande en jeu. Le programme tar est utilisé pour créer des archives de nombreux fichiers, mais il n'a pas ses propres routines de compression. Mais en utilisant les options appropriées avec , nous pouvons faire passer le fichier d'archive à travers . De cette façon, nous obtenons un fichier d'archive compressé et une archive multi-fichiers ou multi-répertoires.

```
$ tar -czvf level1.tar.gz level1
```

```
kayinda@kayinda-VirtualBox:~/Documents/Se2$ tar -czvf level1.tar.gz level1
level1/
level1/level2-2/
level1/level2-2/level3-2-1/
level1/level2-2/level3-2-1/depth3-2-1.txt.gz
level1/level2-2/depth2-2.txt.gz
level1/level2-2/level3-2-2/
level1/level2-2/level3-2-2/depth3-2-2.txt.gz
level1/level2-1/
level1/level2-1/depth2-1.txt.gz
level1/level2-1/level3-1-1/
level1/level2-1/level3-1-1/depth3-1-1.txt.gz
level1/depth1.txt.gz
```

Les options sont les suivantes: tar

- c : Créer une archive.
- z : Poussez les fichiers à travers .gzip
- v : Mode détaillé. Imprimez dans la fenêtre du terminal ce qui est à faire.tar
- f level1.tar.gz : nom de fichier à utiliser pour le fichier d'archive.

```
kayinda@kayinda-VirtualBox:~/Documents/Se2$ tar -czvf level2.tar.gz level2
level2/
level2/depth1.txt
level2/level2-2/
level2/level2-2/level3-2-1/
level2/level2-2/level3-2-1/depth3-2-1.txt
level2/level2-2/depth2-2.txt
level2/level2-2/level3-2-2/
level2/level2-2/level3-2-2/depth3-2-2.txt
level2/level2-1/
level2/level2-1/depth2-1.txt
level2/level2-1/level3-1-1/
level2/level2-1/level3-1-1/depth3-1-1.txt
kayinda@kayinda-VirtualBox:~/Documents/Se2$
```

Cette opération archive la structure de l'arborescence de répertoires et tous les fichiers de l'arborescence de répertoires dans un seul fichier tar.

Nom	Taille	Type	Modifié
level2	0 octet	Dossier	11 sánzá ya mínei 2...

1.2.2. DÉCOMPRESSION

Pour la décompression d'un fichier .gz, nous allons utiliser l'option (décompresser). Cela extraira le fichier compressé de l'archive et le décompressera afin qu'il soit impossible de le distinguer avec le fichier d'origine. -d

Les commandes que nous utilisons pour cet exemple sont:

```
ls fich*
gzip -d fichier.txt.gz
ls fich*
```



```
kayinda@kayinda-VirtualBox:~/Documents/Se$ ls fich*
fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se$ gzip -d fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se$ ls fich*
fichier.txt
kayinda@kayinda-VirtualBox:~/Documents/Se$
```

Une fois de plus, nous remarquons que le fichier d'archive a été supprimé après avoir été extrait le fichier d'origine. Pour conserver le fichier d'archive, nous devons utiliser à nouveau l'option (conserver), ainsi que l'option décompresser `gzip -k -d`

```
kayinda@kayinda-VirtualBox:~/Documents/Se3$ gzip -k -d fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se3$ ls
fichier.txt  fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se3$
```

1.2.2.1. Décompression et écrasement

Si vous essayez d'extraire un fichier dans un répertoire où le fichier d'origine (ou un autre fichier identique) existe, vous serez invité à choisir d'abandonner l'extraction ou de remplacer le fichier existant.

Si vous savez à l'avance que vous souhaitez que le fichier du répertoire soit écrasé par le fichier de l'archive, utilisez l'option `-f` (force).

```
gzip -df fichier.txt.gz
```

```
kayinda@kayinda-VirtualBox:~/Documents/Se3$ ls
fichier.txt  fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se3$ gzip -df fichier.txt.gz
kayinda@kayinda-VirtualBox:~/Documents/Se3$ ls
fichier.txt
kayinda@kayinda-VirtualBox:~/Documents/Se3$
```

2. BZIP2

2.1. PRÉSENTATION

bzip2 est à la fois le nom d'un algorithme de compression de données et d'un logiciel libre développé par Julian Seward entre 1996 et 2000 qui l'implémente. L'extension des fichiers compressés avec ce logiciel est généralement `.bz2`

`bzip2` est très utilisé sous UNIX comme alternative, voire comme remplacement à l'utilitaire `gzip`, du fait de son efficacité supérieure. Il ne le remplace pas totalement car il est significativement moins rapide ; le choix entre les deux logiciels se fait donc selon l'usage et les contraintes.

L'utilisation de **bzip2** comme alternative lente mais efficace à **gzip** est remise en question par l'arrivée de **LZMA**. En effet, cet algorithme encore plus efficace et plus rapide à la décompression

s'impose de plus en plus pour les tâches de distribution (installateurs sous Windows, paquets de certaines distributions de GNU/Linux...).

bzip2 conserve néanmoins des avantages par rapport à **LZMA** : une faible utilisation mémoire à la compression, la robustesse des archives à la corruption et la parallélisation possible sur de nombreux threads (**LZMA** ne peut exploiter plus de deux threads).

2.2. COMPRESSION ET DÉCOMPRESSION AVEC BZIP2

2.2.1. COMPRESSION

Pour créer l'archive via l'algorithme bzip2, nous allons utiliser la commande tar avec plusieurs options, notamment l'option "**j**" pour spécifier l'algorithme bzip2 et l'option "**c**" pour indiquer qu'il s'agit d'une archive à créer.

Voici un exemple pour créer l'archive "MonArchive.tar.bz2" en intégrant dans cette archive "MonFichier1.txt" et "MonFichier2.txt" :

```
tar jcvf MonArchive.tar.bz2 MonFichier1.txt MonFichier2.txt
```

Il est à noter que l'option "-r" ou "--append" peut être utilisée pour ajouter des fichiers à une archive existante.

Passons maintenant à la phase de décompression...

2.2.2. DÉCOMPRESSION

Dès lors que l'on est en possession d'une archive tar.bz2, nous allons utiliser la commande tar avec les options adéquates. L'option "j" est indispensable pour spécifier qu'il s'agit d'une archive compressée via l'algorithme bzip2. Ensuite, nous avons les options classiques notamment "x" pour l'extraction.

Voici un exemple pour extraire le contenu de l'archive "MonArchive.tar.bz2" :

```
tar jxvf /home/MonArchive.tar.bz2
```

Le contenu de l'archive sera extrait dans le dossier courant au niveau du shell Unix. Si l'on veut envoyer le contenu de l'archive vers un autre dossier, par exemple "/tmp/" il suffit de le préciser comme ceci :

```
tar jxvf /home/MonArchive.tar.bz2 /tmp/
```

Si vous obtenez un message d'erreur du type "tar (child): bzip2 : exec impossible: Aucun fichier ou dossier de ce type" lors de l'extraction, c'est qu'il vous manque le paquet "bzip2" sur votre machine. Voici comment l'installer sur CentOS avec yum :

```
yum install bzip2
```

Ensuite, vous pouvez réessayer d'extraire les données. Il ne vous reste plus qu'à exploiter vos données ! Si vous avez besoin d'aide pour utiliser une option supplémentaire, je vous invite à lire la page man de tar.

3. GNU TAR

3.1. PRÉSENTATION

Le programme **tar** (de l'anglais tape archiver, littéralement « archiveur pour bande ») est un logiciel d'archivage de fichiers standard des systèmes de type **UNIX**. Il a été créé dans les premières versions d'UNIX et standardisé par les normes POSIX.1-1988 puis POSIX.1-2001. Il existe plusieurs implémentations tar, la plus couramment utilisée étant **GNU tar**.

3.2. CARACTÉRISTIQUES TECHNIQUE

Un fichier d'archive créé par tar n'est pas compressé. On appelle parfois le fichier d'archivage créé un tarball. L'archivage se fait presque toujours sur un disque. L'usage le plus courant actuellement consiste cependant à créer ou lire un fichier archive. Tar préserve les droits, le propriétaire et le groupe des fichiers et des répertoires. Il permet également de sauvegarder les liens symboliques et les fichiers spéciaux orientés bloc ou caractère.

Il ne compresses pas les fichiers, mais il les concatène au sein d'une seule et même archive. La majorité des programmes linux utilisent ce système d'archivage. Il est aussi souvent utilisé avec le système de compression gzip, donnant alors des archives compressées portant l'extension .tar.gz. Le programme tar est disponible par défaut sous Ubuntu. Il fait partie de l'installation sous linux.

3.3. COMPRESSER ET DÉCOMPRESSER LES FICHIERS DANS UNE ARCHIVE TAR

3.3.1. Gérer les archives .tar avec un logiciel graphique

Le format **tar** (la version en ligne de commande) doit être installé. Il sera utilisé par les logiciels graphiques.

Pour extraire une archive, il suffit de faire un clic-droit sur son fichier (qui sera en **.tar** ou en **.tar.gz**), puis choisir "Extraire l'archive" (ou formulation équivalente selon votre variante d'Ubuntu, comme "Décompresser l'archive").

Pour créer une archive, il suffit de sélectionner les fichiers à compresser dans son explorateur de fichiers, puis faire un clic-droit, "compresser" (ou un équivalent), choisir le .tar ou le .tar.gz dans les formats de compression/archivage proposé, et valider.

Les gestionnaires d'archives ne sont donc pas forcément indispensables, mais votre installation d'Ubuntu en a normalement un, permettant d'aller plus loin dans la manipulation des archives (ajouts ou suppressions partielles, par exemple).

3.3.2. Utilisation en ligne de commandes

Pour tous les formats à base de **tar**, vous verrez que les options de tar sont les mêmes :

- c : crée l'archive
- x : extrait l'archive
- f : utilise le fichier donné en paramètre
- v : active le mode « verbeux » (bavard, affiche ce qu'il fait).

Puis selon la compression souhaitée :

z : ajoute la compression **Gzip**.

j : ajoute la compression **Bzip**.

J : ajoute la compression **Lzma**.

3.3.2.1. Utiliser tar seul: concaténation des fichiers

Création d'une archive, archivage de plusieurs fichiers :

```
tar -cvf archive.tar spence11 fichierarchive2...
```

De même pour un dossier :

```
tar -cvf archivedossier.tar dossier/
```

3.3.2.2. tar : extraction de fichiers

Désarchive et décompresse

```
tar -xvf archivedossier.tar
```

créer le dossier pour décompresser si il n'existe pas

```
mkdir folder
```

Désarchive et décompresse dans un dossier

```
tar -xvf archivedossier.tar -C path_folder
```

3.3.2.3. Compression avec gzip (.tar.gz)

Création

```
$ tar -zcvf votre_archive.tar.gz votre_dossier/
```

Extraction

```
$ tar -zxvf votre_archive.tar.gz
```

```
$ tar -xvzf votre_archive.tar -C path_folder
```

3.3.2.4. Compression avec Bzip2 (.tar.bz2)

Remarques : Bzip crée des fichiers beaucoup plus petits que Gzip, mais utilise plus de ressources processeur surtout pour compresser.

Création

```
$ tar -jcvf votre_archive.tar.bz2 votre_dossier/
```

Extraction

```
$ tar -jxvf votre_archive.tar.bz2
```

3.3.2.5. Compression avec Lzma (.tar.xz)

Ces archives sont des archives Tar compressées avec Lzma, un utilitaire de compression libre parmi les plus puissants : c'est la même méthode de compression que celle utilisée par **7zip**.

Pour utiliser le format « **.xz** », installez le paquet **xz-utils**.

Création :

```
$ tar -jcvf votre_archive.tar.xz votre_dossier/
```

Extraction

```
$ tar -Jxvf votre_archive.tar.xz
```