# SE - TP 3

## Exercice 1 : La gestion du terminal

Seul un processus lancé à partir d'un terminal donné peut être en avant-plan dans ce terminal (i.e. recevoir les commandes qui y sont tapées).

1. Exécutez la commande `gedit`. Pouvez vous taper des caractères dans la fenêtre ouverte par l'application *gedit* ? Dans la fenêtre du terminal ?
2. Sélectionnez le terminal et tapez `ctrl-z`. Pouvez vous agir dans la fenêtre de l'application *gedit* ? et dans le terminal ?
3. A l'aide de la commande `ps -u`, déterminez l'état de *gedit* (colonne STAT). Confirmez cet état avec la commande `jobs`.
4. tapez l'instruction `fg` dans le terminal, puis de nouveau `ps -u`. L'état du processus *gedit* a-t-il changé ? Pouvez vous agir dans la fenêtre de l'application ?
5. Tapez à nouveau `ctrl-z`, puis `bg`, puis de nouveau `ps -u`. L'état de *gedit* a-t-il changé ? Pouvez vous agir dans la fenêtre de l'application ?
6. Fermez l'application *gedit* ; executez maintenant `gedit &`. Pouvez vous agir dans la fenêtre de l'application ? Dans le terminal ? Dans quel état se trouve le processus *gedit* ?

## Exercice 2 : Et si on communiquait avec les processus ?

1. Que fait la commande `evince` ?
2. Exécutez l'application `evince` en arrière plan.
3. Utilisez la commande `ps` pour déterminer l'identifiant (PID) du processus *evince* que vous avez lancé. Appelons ce PID : le pid_de_evince.
4. Tapez la commande `kill pid_de_evince`. Que se passe-t-il ? Pourquoi ?
5. Déterminez le PID d'une commande intitulée *bash* et arrétez-la avec un `kill -9`. Pourquoi la fenêtre du terminal disparait-elle ?
6. Dans un terminal, les processus en avant-plan sont un cas particulier. Ils peuvent être interrompus en tapant `ctrl-c` dans ce terminal. Testez cette séquence de touches en interrompant l'exécution d'un programme `evince` lancé en avant-plan.

## Exercice 3 : Archivage...

```
perso
|-- audio
|-- humour
|   |-- blagues
|   |   |-- histoire1
|   |   `-- histoire2
|   `-- sketch
`-- photos
    |-- photo1
    `-- photo2
```

Les fichiers *histoire*1, *histoire*2, *photo*1 et *photo*2 sont des fichiers de données non vides.

1. Construisez dans votre répertoire personnel l'arborescence ci-dessus.

   Il peut arriver de vouloir regrouper plusieurs fichiers en un seul, par exemple pour l'envoi de piéces jointes par courrier électronique, ou pour la mise à disposition d'un ensemble de fichiers sur internet. On dispose pour cela d'un outil appelé `tar` (pour tape archive, cet outil était auparavant destiné à l'archivage sur bande magnétique). On utilise le suffixe $.tar$ pour distinguer les archives.

   Exemples d'utilisation :

   ```
   tar -cvf toto.tar fic1 fic2 ... : archive fic1,fic2... dans toto.tar
   tar -xvf toto.tar : extrait dans le répertoire courant les fichiers
   contenus dans toto.tar
   ```

2. Archivez les fichiers $histoire1$ et $histoire2$ dans une archive que vous nommerez $testHistoire.tar$.
3. Examinez le contenu de l'archive à l'aide de la commande `cat`. Que constatez-vous ?
4. Désarchivez les fichiers dans un nouveau répertoire $perso2$ crée dans votre répertoire personnel.

   La commande `tar` permet également d'archiver des répertoires entiers directement.

5. Créez une archive $perso.tar$ contenant l'intégralité du répertoire $perso$ précédemment créé.
6. Examinez son contenu à l'aide de la commande `cat`, même si cela reste lisible, cela commence à être inefficace de lire directement le fichier archive.
7. Cherchez dans la page de manuel de `tar` l'option permettant d'examiner le contenu d'une archive. Testez votre commande.

## Exercice 4 : Et compression...

Les fichiers produits à l'aide de la commande `tar` étant parfois trés gros, il est souvent utile de les compresser avant de les envoyer. C'est le but de la commande `gzip`. La commande réciproque `gunzip` permet de décompresser. Cette commande s'attend à ce que les fichiers compressés aient une extension $.gz$ ou $.tgz$ (s'il s'agit d'une archive compressée, comme abréviation de $tar.gz$).

1. Compressez l'archive *perso.tar* précédemment créee. Décompressez l'archive *perso.tar.gz* puis extrayez son contenu dans le répertoire *perso2*.
2. Au lieu de taper deux commandes distinctes, la commande `tar` possède une option permettant de compresser l'archive. En une seule ligne archivez puis compressez les fichiers *histoire1* et *histoire2*.
3. Examinez le contenu de l'archive compressée pour vérifier qu'elle contient bien les fichiers voulus.
4. Il existe aussi les commandes `bzip2` et `bunzip2`. Compresser l'archive *perso.tar*.
5. Le taux de compression est-il meilleur avec cette nouvelle commande ?
6. Essayez de compresser le fichier *perso.tar.gz* avec `bzip2`. Que constatez-vous concernant la taille de l'archive compressée ?
7. Décompressez l'archive en tapant une seule commande, puis essayer de la compresser en faisant le contraire (bzip2 puis gzip). Que constatez-vous ?

   Les outils de messagerie modernes ont des limites de quelques Mo (1 à 5 en général) par message. La commande `split` permet de découper un fichier qui serait toujours trop volumineux en plusieurs morceaux.

8. Donnez la syntaxe de la commande permettant de découper un fichier en morceau de $100ko$ et préfixé par *archiveSrc-* ?

## Exercice 5 : Au pays des tubes !

1. Créez un fichier *fic*. Est-il possible, en utilisant 2 terminaux de lire instantanément dans le premier ce qu'on écrit dans *fic* à partir du deuxiéme ? Pourquoi la commande `cat` se termine-t-elle après avoir imprimé le contenu du fichier ?
2. Créez un fichier nommé *tube1* avec la commande `mkfifo`. Regardez ses caractéristiques (droits, taille, type de fichier, etc.), que constatez-vous ?
3. Essayez d'ouvrir ce fichier dans `gedit`, que constatez-vous ?

   Dans la suite, toutes les lectures et écritures dans les différents fichiers se feront à l'aide de la commande `cat`.
4. Ecrivez dans ce fichier sans terminer la commande.
5. Ouvrez un deuxiéme terminal et lisez le fichier *tube1*. Que constatez-vous ? Essayez d'écrire autre chose dans la commande de la question précédente. Que se passe-t-il ?
6. Que constatez-vous lorsque vous terminez la commande qui écrit dans le tube ? Que pouvez-vous en déduire pour la question 1 ?
7. On appelle généralement écrivain le programme chargé d'écrire dans le tube et lecteur celui qui lit son contenu. Peut-on avoir plusieurs écrivains pour un lecteur ? Essayez avec au moins deux écrivains.
8. Que se passe-t-il si on a plusieurs lecteurs ? Essayez avec un seul écrivain et au moins deux lecteurs. Qu'en déduisez-vous sur le fonctionnement des tubes nommés ?

## Exercice 6 : Et si on tchattait ?

1. Créez un tube nommmé *tube_nomLogin* (ou $nomLogin$ est à remplacer par votre login) à la racine de votre répertoire personnel.
2. Quels sont les droits à affecter à votre répertoire personnel ainsi qu'au fichier *tube_nomLogin* afin de laisser les étudiants du même groupe que vous lire le contenu du tube ? Effectuez ces modifications de droits.
3. Ouvrez 2 terminals et à l'aide de votre voisin effectuez les opérations suivantes :
   (a) Dans le premier terminal, vous jouerez le rôle d'écrivain. A l'aide de la commande `cat > tube_nomLogin` écrivez du texte.
   (b) Dans le deuxiéme terminal, vous jouerez le rôle de lecteur. A l'aide de la commande `cat`, visualisez le contenu du tube **de votre voisin**.
   (c) Demandez à votre voisin d'effectuer les mêmes opération que vous. Que se passe t'il ?