Vous ne voyez que la première page (page écran) et vous avez un *prompt* pour entrer des commandes. **Indispensables :** `q` ou `Ctrl-C` pour quitter, `↩` pour afficher la ligne suivante, `space` pour la page suivante. **Utiles :** `b` pour la page précédente, `/` ou `?` pour des recherches et `h` (*help*) pour l'aide.

28. Affichez `extrait.txt` avec `less` et lancez la recherche de la chaîne `Labesse`. Vous pouvez chercher l'occurrence suivante avec la commande `n` (*next*).
    Ça peut servir, non ? Surtout que comme on l'a déjà vu, la commande `less` est utilisée pour afficher les manuels, combinée avec `man`.
29. Affichez en format long le contenu du répertoire `/usr/bin`. Vous constatez que vous ne voyez pas les premiers fichiers...

    ```
    ls -l /usr/bin
    ```

30. Comme la plupart des commandes Unix, `less` peut être utilisée comme un filtre, c'est-à-dire qu'en l'absence de fichiers à afficher, elle affiche son entrée standard (le clavier par défaut). C'est bien entendu sans intérêt si l'on ne redirige pas l'entrée standard.
    Créez un tube de la commande précédente vers la commande `less`, vous pouvez maintenant voir le résultat page par page.

    ```
    ls -l /usr/bin | less
    ```

31. la commande `grep` effectue la recherche d'une chaîne dans un fichier. Cherchez la chaîne `Labesse` dans `extrait.txt`.

    ```
    grep Labesse extrait.txt
    ```

32. Comme `less`, `grep` peut être utilisée comme un filtre. En combinant `ls` et `grep` avec un tube, faites afficher en format long les fichiers de `/usr/bin` dont le nom contient la chaîne `conv`.

    ```
    ls -l /usr/bin | grep conv
    ```

33. Pratique pour l'illustration, la commande précédente est cependant inutilement compliquée puisqu'on peut utiliser le mécanisme de substitution du shell et la seule commande `ls` pour obtenir le même résultat.

    ```
    ls -l *conv*
    ```

Compte-rendu

Vous rendrez à l'enseignant un exemplaire du sujet complété à la fin du TP. Si vous n'avez pas fini, utilisez l'exemplaire qui vous reste pour terminer pendant votre temps libre.

FAUGERON TYLAN
CHALENCON YOAN

19,5/20

Utilisation du shell Unix
4 - Substitutions et redirections

Rappel : vous devez simplement effectuer le travail demandé et noter les commandes utilisées et les réponses aux questions dans l'espace laissé libre. **Attention :** l'objectif est que vous soyez capables de refaire SEUL les manipulations décrites.

Pré-requis

Vous devez avoir parcouru les chapitres « Le langage de commande » et « Les processus » dans les diapos de cours : R1.04-IntroSysteme/Unix.pdf

1 Substitutions simples

1. Vous avez en principe un répertoire `mesTPs/Systeme` dans votre répertoire de travail par défaut (*HOME directory*). Placez-vous dans ce répertoire.

   ```
   cd mesTPs/Systeme
   ```

2. Récupérez l'archive `/users/but/info/Public/shell.tgz`

   ```
   cp /users/but/info/Public/shell.tgz .
   ```

   Puis extrayez-la avec la commande :

   ```
   tar zxf shell.tgz
   ```

   Cela doit créer un répertoire `shell`.
   NB : on aurait pu aussi extraire directement l'archive dans le répertoire courant (sans la copier) avec :

   ```
   tar zxf /users/but/info/Public/shell.tgz
   ```

3. Un bon moyen d'inspecter le contenu d'un répertoire est d'afficher *récursivement* son contenu avec `ls`.

   ```
   ls -R
   ```

   L'affichage récursif de `ls` n'est pas très lisible, utilisez plutôt :

   ```
   tree   et   tree -d
   ```

Placez vous dans le répertoire `shell/noms`.

4. Un seul des fichiers de ce répertoire n'est pas vide, déterminer lequel (utiliser `ls`).

   ```
   ls -l   => le fichier non vide est cr.odt
   ```

5. Faites afficher tous les fichiers dont le nom se termine par `.c`.

   ```
   ls -a *.c
   ```

6. Trouvez deux méthodes permettant d'afficher le nom de tous les fichiers dont le nom commence par un `a` ou un `g`.

   ```
   ls -a [ag]*   |   ls -a a* g*
   ```

7. Faites afficher le nom de tous les fichiers dont le nom se termine par une extension ne contenant qu'un seul caractère (*i.e.* `tp.c` ou `truc.x`).

   ```
   ls -a *.?
   ```

8. Faites afficher le nom de tous les fichiers dont le nom contient exactement 3 caractères.

```
ls -a ???
```

9. Faites afficher avec `echo` la valeur de la variable `PRINTER` (nom de l'imprimante par défaut).

```
echo $PRINTER
```

10. Donnez la commande qui affiche exactement ce message (au nom de l'imprimante près) :
`J'utilise l'imprimante "Impression_UGA" ($PRINTER=Impression_UGA)`

```
echo J'utilise l'imprimante \"$PRINTER\" \($PRINTER=$PRINTER\)
```

2 Compilation et arrêt de programme

Utilisez le document `MementoUnixIUT.pdf` de l'intranet de D. Genthial pour déterminer les commandes à utiliser pour compiler un programme C en ligne de commande (sans utiliser Geany).

11. Revenez au répertoire `shell` (père de `noms`).

```
cd ..
```

12. Compilez puis lancez l'exécution du programme C contenu dans le fichier `bravo.c` (l'exécutable devra s'appeler `bravo`).

```
gcc -Wall bravo.c -o bravo
bravo
```

13. Compilez le programme du fichier `boucle.c` dans un fichier `boucle` et lancez son exécution. Arrêtez-le avec la touche Ctrl-C.

```
gcc -Wall boucle.c -o boucle
boucle
```

3 Redirigeons un peu

Vous êtes toujours dans le répertoire `shell`.

14. Lancez l'exécution de `boucle` en redirigeant sa sortie standard dans un fichier nommé `boucle.out`.

```
boucle > boucle.out
```

15. Quelle est la taille du fichier `boucle.out` ?

```
ls -l → 23 031 300 octets
```

16. Faites pareil avec `bravo`. Avez-vous des messages sur l'écran ?

```
bravo > bravo.out
Oui, on a le message de la sortie des erreurs (**** Bonjour le monde (sur la sortie des erreurs)!)
```

17. Lancez `bravo` en redirigeant la sortie des erreurs sur un fichier `bravo.err`. Avez-vous encore des messages sur l'écran ?

```
bravo 2> bravo.err
Oui, on a le message de la sortie standard (=== = Bonjour le monde (sur la sortie standard)!)
```

18. Ouvrez `bravo.c` : la lecture du code devrait vous permettre de comprendre son fonctionnement. Expliquez pourquoi dans les deux questions précédentes on avait encore des messages sur l'écran malgré la redirection.

```
Le code affiche un message sur la sortie standard et un sur la sortie des erreurs. Donc, si on ne redirige qu'une des deux sorties, le message de l'autre sortie s'affiche dans le terminal.
```

19. Donnez la commande lançant `bravo` en redirigeant à la fois la sortie normale et la sortie des erreurs :

```
bravo >bravo.out 2> bravo.err
```

20. On peut observer un comportement analogue avec une commande standard. Essayez la commande `ls -l bravo machin`, puis refaites-le en redirigeant d'abord la sortie, puis la sortie des erreurs.

4 Redirigeons encore

La commande `cat` (*catenate*) envoie le contenu des fichiers passés en paramètre sur sa sortie standard. Si aucun fichier n'est passé en paramètre, c'est le contenu de son entrée standard qui est envoyé sur la sortie.

Vous êtes toujours dans le répertoire `shell`.

21. Utilisez `cat` pour faire afficher à l'écran le contenu des fichiers `m1.txt` et `m2.txt` en une seule commande.

```
cat m1.txt m2.txt
```

22. Utilisez `cat` pour créer une copie du fichier `m1.txt` baptisée `m1.bis`.

```
cat m1.txt > m1.bis
```

23. Toujours avec `cat`, **ajoutez** le contenu du fichier `m2.txt` à `m1.bis`.

```
cat m2.txt >> m1.bis
```

24. Vérifiez la commande précédente en affichant le contenu de `m1.bis` (toujours avec `cat`).

```
cat m1.bis
```

25. Utilisez `cat` pour créer un fichier `toto` contenant un texte que vous taperez au clavier (terminez le texte par Ctrl-D qui est la marque de fin de fichier sur Unix).

```
cat -> toto
texte
```

5 Mes premiers tubes

La commande `cat` permet de visualiser le contenu d'un ou plusieurs fichiers textes, mais elle n'est pas très pratique si le texte est long. Vous êtes toujours dans le répertoire `shell`.

26. Affichez le contenu du fichier `extrait.txt` avec `cat`. Si vous voulez voir le début du fichier, vous devez utiliser l'ascenseur de la fenêtre.

```
cat extrait.txt
```

27. Affichez maintenant le fichier avec la commande `less` (ou son équivalent `more`).

```
less extrait.txt
```