System d exploitation
rapport TP2:

Partie 1:

Exercice 1:

1.Des explications :

- Pour créer un fichier sur le terminal on utilise la commande suivante :

  ~touch <nom fichier>~

- On peut vérifier que ce dernier été bien créé grâce à la commande :

  ~ls~ ⇒ affiche tous les fichiers .

- Si on veut écrire dans un fichier on le fait avec la commande cat comme suit :

  ~cat> nom file ~

- Puis écrire ce qu' on veut . Click [CTRL D] pour quitter l' écriture .

- Maintenant pour la création de l arbre de répertoires on a deux méthodes :

Soit :

- On utilise

  ~mkdir TPSYS~ creation of directory

  ou

  ~touch hello.c~creation of files

- Puis la création des deux directory MEM et CPU dans le directory TPSYS :

- ~mkdir TPSYS/CPU ~et ~mkdir TPSYS/MEM ~ puis utiliser touch pour création du file hello.c
  Ou
- Changer le répertoire grâce à ~cd TPSYS~puis mkdir {MEM CPU } puis ~touch hello.c~

```
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
ubuntu@ubuntu:~$ mkdir TPSYS
ubuntu@ubuntu:~$ mkdir TPSYS/MEM
ubuntu@ubuntu:~$ mkdir TPSYS/CPU
ubuntu@ubuntu:~$ touch TPSYS/hello.c
```

Les types de chemins :
Chemin relatif : ~/racine/nom_répertoire ~
Chemin absolu:Un chemin absolu ne dépend pas du répertoire courant et est donc valide partout.

-On fait la même chose pour création des directories processus et schedule et leur contenu

```
~cd CPU ~
~mkdir {processus schedule
}~
touch CPU
/processus/file.c per.c
touch CPU /scheduler /rr.c
fif.c sjf.c
```

- Des captures de la création de l arbre:

```
ubuntu@ubuntu:~$ cd TPSYS
ubuntu@ubuntu:~/TPSYS$ cd CPU
ubuntu@ubuntu:~/TPSYS/CPU$ mkdir processus scheduler
ubuntu@ubuntu:~/TPSYS/CPU$ touch processus/file.c pere.c
ubuntu@ubuntu:~/TPSYS/CPU$ touch scheduler/rr.c fif.c sjf.c
ubuntu@ubuntu:~/TPSYS/CPU$
```

```
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
ubuntu@ubuntu:~$ mkdir TPSYS
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  snap       TPSYS
Documents  Music      Public    Templates  Videos
```

```
ubuntu@ubuntu:~/TPSYS$ ls
CPU  hello.c  MEM
```

```
ubuntu@ubuntu:~/TPSYS/CPU/scheduler$ touch rr.c fif.c sjf.c
ubuntu@ubuntu:~/TPSYS/CPU/scheduler$ ls
fif.c  rr.c  sjf.c
ubuntu@ubuntu:~/TPSYS/CPU/scheduler$
```

```
ubuntu@ubuntu:~/TPSYS/CPU$ cd processus
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls
file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ touch pere.c
```

On utilise:

cd⇒ retour au répertoire de login (dans ce cas ~)

cd/ ⇒ retour au répertoire parent (dans ce cas TPSYS)

cd chemin(absolu or relatif) ⇒ repertoire choisi

cd .. ⇒ retour au répertoire père

```
ubuntu@ubuntu:~/TPSYS/CPU$ cd
ubuntu@ubuntu:~$
```

```
ubuntu@ubuntu:~/TPSYS/CPU/scheduler$ cd ..
```

```
ubuntu@ubuntu:~/TPSYS/CPU$ c
```

2.(J' avais oublié que c'était des fichier type .c et non pas .txt donc il fallait écrire un code en langage C)

```
ubuntu@ubuntu:~$ touch essayer1.c
ubuntu@ubuntu:~$ cat>essayer1.c
j ai creer ce fichier avec touch et je le remplis avec cat >
ubuntu@ubuntu:~$ cat>essayer2.c
pour ce fichier puisque il n existe pas il a ete creer automatiquement  et la
je le remplis avec cat>
ubuntu@ubuntu:~$ cat>TPSYS/hello.c
hellooo ce fichier est deja existant
ubuntu@ubuntu:~$
```

3.On utilise la commande ~ls~ avec path relative (du racine au scheduler) comme suit

ls ~/TPSYS/CPU/scheduler~

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls ~/TPSYS/CPU/scheduler
fif.c  rr.c  sjf.c
```

4.Création d' un nouveau répertoire avec mkdir

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ mkdir ~/TPSYS/progc
```

Puis avec move on déplace file hello.c a progc

La commande :

mv ⇒ move (déplacer un file)

mv source destination

Donc on l' utilise comme suit :

```
ubuntu@ubuntu:~/TPSYS$ mv hello.c  progc/hello.c
ubuntu@ubuntu:~/TPSYS$ ls
CPU  MEM  progc
ubuntu@ubuntu:~/TPSYS$ cd progc
ubuntu@ubuntu:~/TPSYS/progc$ ls
hello.c
```

5.Pour copier des fichier existant dans un répertoire et les mettre dans un autre répertoire existant on utilise la commande cp

~cp nom de fichier /destination~

Par défaut, cp n'affiche pas les fichiers lors de la copie.
Pour forcer l'affichage, il faut utiliser le paramètre -v.

~cp -v source destination~

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ cp -v  * ~/TPSYS/progc
'file.c' -> '/home/ubuntu/TPSYS/progc/file.c'
'pere.c' -> '/home/ubuntu/TPSYS/progc/pere.c'
```

Soit donc :

Pour copier tous les fichiers du répertoire processus vers progc on fait :

~cp * /destination/~

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ cp * ~/TPSYS/progc
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls  ~/TPSYS/progc
file.c  hello.c  pere.c
```

6.Supprimer le fichier hello.c

```
ubuntu@ubuntu:~$ rm ~/TPSYS/progc/hello.c
ubuntu@ubuntu:~$ ls ~/TPSYS/progc
file.c  pere.c
ubuntu@ubuntu:~$
```

Pour supprimer tous les fichier .c ou .txt dans un répertoire on fait :~rm *.txt~

7.Le répertoire MEM est vide alors que scheduler se contient de fichiers donc si on utilise rmdir :
sur le répertoire MEM

```
ubuntu@ubuntu:~$ rmdir ~/TPSYS/MEM
ubuntu@ubuntu:~$ ls ~/TPSYS
CPU  progc
ubuntu@ubuntu:~$
```

Sur scheduler:

```
ubuntu@ubuntu:~$ rmdir ~/TPSYS/CPU/scheduler
rmdir: failed to remove '/home/ubuntu/TPSYS/CPU/scheduler': Directory not empty
```

donc IL faut supprimer the directory avec la commande suivante :récursivement: ~rm -r directory name ~or forcement : ~rm -rf directory name ~

```
ubuntu@ubuntu:~$ rm -rf ~/TPSYS/CPU/scheduler
ubuntu@ubuntu:~$ ls ~/TPSYS/CPU
pere.c  processus
```

supprimer scheduler avec son contenu

Exercice 2:

Il y a quatre catégorie d utilisateur sous linux:

1. USER (propriétaire) ⇒ u
2. GROUP (groupe) ⇒ g
3. OTHERS(autres) ⇒ o

on peut avoir tous (ALL) ⇒ a

Les droits attribués sont : Lecture (r⇒read 4) ,Ecriture (w⇒write 2),Execution(x⇒exécute 1).
Représentation symbolique : (r,w,x) or Représentation Octale ⇒ références aux droits .

u: rwx g: r-x o: r- - (par defaut) donc u :somme de 4+2+1

1.Home directory :

```
ubuntu@ubuntu:~$ cd ..
ubuntu@ubuntu:/home$
```

2.Les droits d'accès au file père.c in progc sont:
Lecture et écriture

-6-6-4- Donc u:rw g:rw o:r only

```
ubuntu@ubuntu:/home$ ls -l ~/TPSYS/progc/pere.c
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 18:29 /home/ubuntu/TPSYS/progc/pere.c
```

3-Changer les droits d'accès of directory progc a :- -x - - - - - -

```
ubuntu@ubuntu:/home$ chmod 100 ~/TPSYS/progc
ubuntu@ubuntu:/home$ ls -l ~/TPSYS
total 0
drwxrwxr-x 3 ubuntu ubuntu 80 May 29 19:10 CPU
d--x------ 2 ubuntu ubuntu 80 May 29 18:48 progc
```

Mais si on veut ouvrir le fichier ou écrire dans le fichier on ne peut pas puisque on a enlever les droits(r/w)
On peut ajouter les permission avec +x -rw (avec des + et - et toujours en utilisant la commande chmod)

```
ubuntu@ubuntu:/home$ ls -l ~/TPSYS/progc
ls: cannot open directory '/home/ubuntu/TPSYS/progc': Permission denied
```

Donc :pour y remédier on fait : ajouter le droit de lecture au user : chmod 500 (read +execute)

```
ubuntu@ubuntu:/home$ chmod +r ~/TPSYS/progc
ubuntu@ubuntu:/home$ ls -l ~/TPSYS/progc
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 18:29 file.c
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 18:29 pere.c
```

```
ubuntu@ubuntu:~/TPSYS$ ls -l
total 0
drwxrwxr-x 3 ubuntu ubuntu 80 May 29 19:10 CPU
dr-x------ 2 ubuntu ubuntu 80 May 29 18:48 progc
```

Maintenant on voit que user sur progc a les droits exécute et lecture

5.Non on ne peut pas écrire ,pour écrire we add écriture avec:

```
ubuntu@ubuntu:/home$ touch ~/TPSYS/progc/helloagain.c
touch: cannot touch '/home/ubuntu/TPSYS/progc/helloagain.c': Permission denied
```

```
ubuntu@ubuntu:/home$ chmod +w ~/TPSYS/progc
ubuntu@ubuntu:/home$ touch ~/TPSYS/progc/helloagain.c
ubuntu@ubuntu:/home$ ls -l ~/TPSYS/progc
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 18:29 file.c
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 21:25 helloagain.c
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 18:29 pere.c
ubuntu@ubuntu:/home$ ls -l ~/TPSYS
total 0
drwxrwxr-x 3 ubuntu ubuntu  80 May 29 19:10 CPU
drwxrw-r-- 2 ubuntu ubuntu 100 May 29 21:25 progc
```

6.Révoquer le droit d'exécution au répertoire processus
se mettre dans le répertoire CPU /

```
ubuntu@ubuntu:/home$ cd
ubuntu@ubuntu:~$ cd ~/TPSYS/CPU
ubuntu@ubuntu:~/TPSYS/CPU$
```

```
ubuntu@ubuntu:~/TPSYS/CPU$ cd processus
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls
file.c  pere.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ cd ..
```

puis enlever le droit executer (-x):

```
ubuntu@ubuntu:~/TPSYS/CPU$ cd processus
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 May 29 17:23 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ cat file.c
#include <stdio.h>

int main(){
    printf("hello world");
    return 0;
}
```

ça affiche le contenu du fichier (fils.c (file.c chez moi)), ON a pas besoin d'exécuter le file pour afficher son contenu

8. On crée un fichier test

```
ubuntu@ubuntu:~$ cd ~/TPSYS/CPU
ubuntu@ubuntu:~/TPSYS/CPU$ touch essayer8.c
```

On révoque le droit de lecture et ajoute que écriture

```
ubuntu@ubuntu:~/TPSYS/CPU$ chmod +w-r essayer8.c
ubuntu@ubuntu:~/TPSYS/CPU$ cat>essayer8.c
#include<stdio.h>
int main(){
printf("hello essayer");
return 0;
}^C
ubuntu@ubuntu:~/TPSYS/CPU$ cat essayer8.c
cat: essayer8.c: Permission denied
ubuntu@ubuntu:~/TPSYS/CPU$
```

On remarque qu' on peut juste écrire dans le fichier on ne peut pas lire ni : on ne peut pas l'exécuter non plus il nous faut la lecture

```
ubuntu@ubuntu:~/TPSYS/CPU$ ./essayer8.c
bash: ./essayer8.c: Permission denied
ubuntu@ubuntu:~/TPSYS/CPU$ ls -l  essayer8.c
--w--w---- 1 ubuntu ubuntu 65 May 29 21:33 essayer8.c
ubuntu@ubuntu:~/TPSYS/CPU$ chmod +x essayer8.c
ubuntu@ubuntu:~/TPSYS/CPU$ ./essayer8.c
bash: ./essayer8.c: Permission denied
ubuntu@ubuntu:~/TPSYS/CPU$
```

9.avec la représentation octale : the file is fils.c mais chez moi je l' ai créé file.c (je m excuse madame)

commande	sa signification	liste des droits
chmod 666 file.c	on attribue les droits au user-groupe-others 6-6-6	6=2+4=write+read donc les droits attribués sont lecture et écriture
chmod 751 file.c	on attribue les droits au user-groupe-others 7-5-1	7=1+2+4=write +read +execute donc user a tous les droits 5=4+1 donc groupe a les droits suivante:read and execute

chmod 234 file.c	on attribue les droits au user-groupe-others 2-3-4	2=write user can only write 3=2+1=write+execute groupe can only do that 4=read others can only write
chmod g+w file.c	on ajoute l'écriture sur le fichier file.c a la catégorie groupe	g+w donc ca sera 3+w mais 3=exécute and write donc c est la meme chose il y aura le droit d écriture
chmod g-rx,o+rw file.c	on enlève les droits read and execute from group category and ajouter read and write au autres	/
chmod u-rw	on enleve les droits read and write from user	/user n a aucun droit
chmod a=r	all = read	/tous les utilisateurs ont le droit de lecture seulement

10-traduction :

commande	sa conversion
chmod 666 file.c	chmod a=rw file.c
chmod 751 file.c	chmod u=rwx,g=rx,o=x file.c
chmod 234 file.c	chmod u=w,g=wx,o=r file.c
chmod g+w file.c(reste same puisque g=wx already	(avant exécution on a -2-3-4) chmod 234 file.c
chmod g-rx,o+rw file.c	(avant exécution on a -2-3-4) chmod 226 file.c

chmod u-rw file.c	(avant exécution on a -2-2-6) chmod 026 file.c
chmod a=r file.c	(avant exécution on a -0-2-6) chmod 444 file.c

11-exécution:

Avant exécution des commandes :

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
-rw-rw-r-- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

Après exécution de chaque commande

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod 666 file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
-rw-rw-rw- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod 751 file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
-rwxr-x--x 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod 234 file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
--w--wxr-- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod g+w file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
--w--wxr-- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod g-rx,o+rw file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
--w--w-rw- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod u-rw file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
-----w-rw- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

```
ubuntu@ubuntu:~/TPSYS/CPU/processus$ chmod a=r file.c
ubuntu@ubuntu:~/TPSYS/CPU/processus$ ls -l
total 4
-r--r--r-- 1 ubuntu ubuntu 78 May 29 21:12 file.c
```

12.Modification sur le répertoire /dev :
On exécute la commande ~chmod a=r /dev~
 result:

```
ubuntu@ubuntu:~$ chmod a=r /dev
chmod: changing permissions of '/dev': Operation not permitted
```

/dev est un Répertoire(dont le propriétaire est le root) qui contient les périphériques , on trouve les caractère c et b :
c⇒type character ,b⇒ bloc
ex: sda est un disque de type bloc
t⇒ sticky bit ca concerne seulement les utilisateurs others (répertoire temporaire),pas droit de toucher les répertoires d' autre utilisateurs

La raison:
On a pas le droit de modifier les répertoires du super utilisateur (Root),que root peut modifier enlever ou ajouter des droits à ce type de répertoire .

- Par contre on peut consulter ce dernier avec ls et autre commandes
- On peut le faire avec :

```
ubuntu@ubuntu:~$ sudo chmod a=rwx /dev
ubuntu@ubuntu:~$
```

```
drwxrwxrwx  18 root root 3980 May 29 15:59 dev
```

13.La commande umask:est un C-shell built command qui permet de déterminer et spécifier l'accès par défaut pour le nouveau fichier qu' on crée .

Executing ~umask~ affiche un numéro: de Quatre bits :

```
ubuntu@ubuntu:~$ umask
0002
```

0⇒superuser;0⇒ user;0⇒group;2⇒others(root).

14.Creation du repertoire et son contenu:

```
ubuntu@ubuntu:~$ mkdir interruption
ubuntu@ubuntu:~$ touch interruption/signal.c
```

15.Affichage des droits par défauts au répertoire interruption:

```
drwxrwxr-x 2 ubuntu ubuntu 60 May 30 00:10 interruption
```

```
ubuntu@ubuntu:~$ ls -l ~/interruption
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 May 30 00:10 signal.c
```

16.Exécution de la commande ~umask 026~

```
ubuntu@ubuntu:~$ umask 026
```

```
ubuntu@ubuntu:~$ umask
0026
```

17.Création du nouveau fichier :

```
ubuntu@ubuntu:~$ touch signal1.c
```

Ses droits:

```
-rw-r----- 1 ubuntu ubuntu  0 May 30 00:14 signal1.c
```

Remarque +explications:

- Les droit par défaut :

rw for all ; mais on créant le fichier on trouve qu' il y a des droits qui ont étaient enlever,on appliquant le masque par défaut (mask)

0⇒ enlever rien 7-0
0⇒ enlever rien aussi pour user 6-0
2⇒ enlever l écriture w=2 pour group donc 6-2=4=read
6⇒ enlever tous 6-6=0 droits

Partie 2:

Exercice 3 (Redirections et pipes):

Une commande peut

- **lire les données à traiter à partir d'un fichier et non du clavier de l'utilisateur, et**
- **écrire les résultats ou erreurs dans un fichier et non à l'écran.**

1.Se placer sur Home :

```
ubuntu@ubuntu:~$ cd ..
ubuntu@ubuntu:/home$
```

2.On fait la redirection du l erreur vers un fichier "erreur" et pas la sortie standard ;en précisant que le message sera redirigé en cas d erreur ,et puis on affiche son contenu :

```
ubuntu@ubuntu:~$ ls titi 2> erreur.txt
ubuntu@ubuntu:~$ cat erreur.txt
ls: cannot access 'titi': No such file or directory
```

3. part 1: exécution de la commande ~ls titi * 2> erreur.txt ~ ⇒ elle affiche tous les documents ainsi que leurs contenus

capture :

le message erreur n' a pas été affiché

```
ubuntu@ubuntu:~$ ls titi * 2>erreur.txt
erreur.txt  essayer.c  signal1.c

Desktop:
ubiquity.desktop

Documents:

Downloads:
Ubuntu Software
interruption:
signal.c

Music:

Pictures:

Public:

snap:
snapd-desktop-integration

Templates:

TPSYS:
CPU  progc
```

Remarque :

Si on affiche le contenu d erreur.txt : on trouve le même contenu de la deuxième question ; parce que on préciser que le texte sera enregistré sur erreur que si il y a des erreurs

part 2: faire la même chose juste changer le fichier a resultat.txt :~ls titi * > resultat.txt ~

maintenant on stock tous l' affichage dans le fichier résultat sauf l erreur:

le résultat:

```
ubuntu@ubuntu:~$ ls titi * >resultat.txt
ls: cannot access 'titi': No such file or directory
ubuntu@ubuntu:~$ cat resultat.txt
erreur.txt
essayer.c
resultat.txt
signal1.c

Desktop:
ubiquity.desktop

Documents:

Downloads:
```

Note : > pour diriger le contenu vers le fichier précis , alors que 2> pour diriger le message erreurs vers le fichier choisi.(>> c est un ajout au fichier)

4.faire la même chose mais rediriger les deux (erreur ,contenu) vers le même fichier

on veut rediriger la sortie standard ainsi que la sortie d' erreur vers ce fichier (res_err.txt)

on le fait avec la commande suivante

~ls titi * > res_err.txt 2>&1~

```
ubuntu@ubuntu:~$ ls titi * > res_err.txt 2>&1
ubuntu@ubuntu:~$
```

Rien n' est affiché ni le résultat ni l' erreur pour le fichier inexistant titi : ⇒ cause: tout est dirigé vers le fichier res_err.txt.

affichage de son contenu::

```
ubuntu@ubuntu:~$ cat res_err.txt
ls: cannot access 'titi': No such file or directory
erreur.txt
essayer.c
resultat.txt
signal1.c

Desktop:
ubiquity.desktop

Documents:

Downloads:

interruption:
signal.c

Music:

Pictures:

Public:

snap:
snapd-desktop-integration
```

5. Pour afficher le manuel de mkdir on utilise la commande ~man mkdir~
et pour rediriger le fichier de sortie standard on utilise la commande :
~man mkdir > manuel ~

```
ubuntu@ubuntu:~$ man mkdir >manuel
```

6.avec ~cat manuel ~
on affiche le contenu

```
MKDIR(1)                        User Commands                        MKDIR(1)

NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION]... DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory  arguments  to  long options are mandatory for short options
       too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed

       -v, --verbose
              print a message for each created directory

       -Z     set SELinux security context of each created directory  to  the
              default type

       --context[=CTX]
              like  -Z,  or if CTX is specified then set the SELinux or SMACK
              security context to CTX
```

7.
On recopie le contenu de manuel vers manuelbis avec la commande suivante :

```
ubuntu@ubuntu:~$ cat manuel > manuelbis
```

si on exécute la commande :~cat manuelbis ~ elle affiche le même résultat que manuel (c'est une copie)

8. En exécutant la commande ~ man mkdir >manuel >manuelbis~ le contenu du manuel mkdir va être redirigé vers manuel d'abord puis déplacer (couper et non copier) vers manuelbis

9. Pour ajouter au fichier manuelbis, le manuel de la commande rmdir, on fait:
~man rmdir >> manuelbis ~
10. Pour afficher les attributs de manuelbis dans un fichier droits:
~ls -l manuelbis > droits ~

```
Saras-Mac:~ mbpro$ ls -l manuelbis > droits
Saras-Mac:~ mbpro$ cat droits
-rw-r--r--  1 mbpro  staff  3738 Jun  1 19:38 manuelbis
```

11. afficher à l'écran le contenu des deux fichiers manuelbis et droits en une seule commande:

~cat manuelbis droits~

<u>En utilisant le symbol tube:</u>

12. En utilisant les pipes et en une seule commande, affichez page par page la liste triée de tous les fichiers du répertoire courant avec leurs attributs:

~ ls -l | sort | more ~

11. Ouvrez une deuxième console: ~gnome-terminal ~

Vérifiez son numéro par la commande tty:

Transférez l'affichage du résultat précédent sur la deuxième console:

~ tty > /dev/pts/1 ~

~ cat erreur.txt >/dev/pts/1~

12. rediriger le résultat de la question 12 vers /dev/null ? Que se passe-t-il ?

commande:

~ tty > /dev/null ~

cette commande sert à ne pas afficher aucun résultat car on l'a pas besoin dans quelques traitements

Si on affiche avec:

~cat /dev/null ~

Rien ne se passe car ce fichier est un fichier puis, utilisé pour se débarrasser d'un résultat d'une commande.

<u>Ps</u>*:* La commande echo $? sert a affiche le code de retour du dernière commande (pour savoir si elle était effectué avec succès ou échec dans le cas ou on déplace le résultat vers dev/null)

(zero ---> succès , autre ---> échec).

Exercice 4 (Liens) :

-pour un lien symbolique:
ln -s nomfich nomlien
-pour un lien physique:
ln nomfich nomlien

1. Créez un fichier test contenant la ligne Premier fichier de test. Créez une copie de ce fichier, test1, puis un lien physique test2 et un lien symbolique test3 sur test.

```
[Saras-Mac:~ mbpro$ touch test
[Saras-Mac:~ mbpro$ cat> test
 Premier fichier de test.
[Saras-Mac:~ mbpro$ cat test
 Premier fichier de test.
[Saras-Mac:~ mbpro$ cp test test1
[Saras-Mac:~ mbpro$ ln test test2
[Saras-Mac:~ mbpro$ ln -s test test3
 Saras-Mac:~ mbpro$
```

2.Comparez les numéros d'inodes de ces quatre fichiers. (L'option -i de ls permet d'afficher le numéro d'inode)

```
Saras-Mac:~ mbpro$ ls -i -l test test1 test2 test3
12895592308 -rw-r--r--  2 mbpro  staff  25 Jun  1 05:47 test
12895596425 -rw-r--r--  1 mbpro  staff  25 Jun  1 17:18 test1
12895592308 -rw-r--r--  2 mbpro  staff  25 Jun  1 05:47 test2
12895596485 lrwxr-xr-x  1 mbpro  staff   4 Jun  1 17:19 test3 -> test
Saras-Mac:~ mbpro$
```

3.Supprimez test, puis affichez le contenu de test1, test2, test3. Que constatez-vous ?

```
Saras-Mac:~ mbpro$ rm test
Saras-Mac:~ mbpro$ cat test1
premier fichier de test.
Saras-Mac:~ mbpro$ cat test2
premier fichier de test.
Saras-Mac:~ mbpro$ cat test3
cat: test3: No such file or directory
Saras-Mac:~ mbpro$
```

test3 est introuvable car c'est un raccourci de test.

4.Créez un nouveau fichier test contenant le texte Deuxième fichier de test. Affichez à nouveau le contenu des quatre fichiers. Que constatez-vous ?

```
Saras-Mac:~ mbpro$ touch test
Saras-Mac:~ mbpro$ cat> test
Deuxieme fichier de test .
Saras-Mac:~ mbpro$ cat test
Deuxieme fichier de test .
Saras-Mac:~ mbpro$ cat test1
premier fichier de test.
Saras-Mac:~ mbpro$ cat test2
premier fichier de test.
Saras-Mac:~ mbpro$ cat test3
Deuxieme fichier de test .
Saras-Mac:~ mbpro$
```

on remarque que le fichier test3 va contenir le contenu du test car c'est son raccourci
les fichiers test1 vont contenir la phrase "premier fichier de test" car ce sont des copies du premier fichier texte supprimé

Exercice 5 (recherche):

1.Dans votre répertoire courant, créez en une seule commande les fichiers annee1,Annee2, année 4, année 45, année 410, annee510, et annee_lunaire.

```
Saras-Mac:~ mbpro$ touch annee1 Annee2 annee4 annee45 annee410 annee510 annee_lunaire
Saras-Mac:~ mbpro$ mkdir ans
Saras-Mac:~ mbpro$ mkdir nouvelAns
```

2. Créez les répertoires Ans et NouvelAns dans votre répertoire courant, et en une seule commande déplacez les fichiers précédemment créés dans le répertoire Ans.

```
Saras-Mac:~ mbpro$ mkdir ans
Saras-Mac:~ mbpro$ mkdir nouvelAns
```

supposant que c'est les seuls fichiers qui commencent par a ou A

```
Saras-Mac:~ mbpro$ mv [aA]* Ans
Saras-Mac:~ mbpro$ ls -a ans
.               Annee2          annee1          annee410        annee510
..              Applications    annee4          annee45         annee_lunaire
```

3.Listez les fichier:

on fait d'abord cd ans après:

Qui se termine par 5:

```
Saras-Mac:ans mbpro$ ls *5
annee45
```

Qui commencent par année4:

```
Saras-Mac:ans mbpro$ ls annee4*
annee4          annee410        annee45
```

Qui commencent par année4 et de 7 lettres au maximum:

```
Saras-Mac:ans mbpro$ ls annee4?
annee45
```

Qui commencent par année avec aucun chiffre numérique:

```
Saras-Mac:ans mbpro$ ls annee[!0-9]*
annee_lunaire
```

Qui contiennent la chaîne una:

```
Saras-Mac:ans mbpro$ ls *una*
annee_lunaire
```

:

Qui commencent par a ou A:

```
Saras-Mac:ans mbpro$ ls [aA]*
Annee2          annee4          annee45         annee_lunaire
annee1          annee410        annee510
```

4.Copiez les fichiers dont l'avant dernier caractère est un 4 ou 1 dans le répertoire NouvelAns en une seule commande.

D'abord on se positionne dans le répertoire ans:

cd ans

après on fait: cp *[14]? ../nouvelAns

Exercice 6:

Archivage:

La commande :

```
$ tar cvf rep.tar rep
```

permet de créer un fichier d'archive **rep.tar** à partir du répertoire **rep**.

La commande :

```
$ tar tvf rep.tar
```

permet d'obtenir la liste des fichiers archivés.

La commande :

```
$ tar xvf rep.tar
```

permet d'extraire le contenu de l'archive **rep.tar** dans le répertoire courant.

Créez une archive TPSYS.tar du répertoire TPSYS,
Créez un répertoire Archive et placez-y votre fichier d'archive,
Désarchiver votre fichier à l'intérieur du répertoire Archive.
D'abord mkdir tpsys après:

```
[Saras-Mac:~ mbpro$ tar cvf tpsys.tar tpsys
a tpsys
[Saras-Mac:~ mbpro$ mkdir archive
[Saras-Mac:~ mbpro$ mv tpsys.tar archive
[Saras-Mac:~ mbpro$ cd archive
[Saras-Mac:archive mbpro$ tar xvf tpsys.tar
x tpsys/
Saras-Mac:archive mbpro$
```

Compression:

Créez une archive compressée TPSYS.tar.gz du répertoire TPSYS.

```
[Saras-Mac:archive mbpro$ gzip tpsys.tar
[Saras-Mac:archive mbpro$ ls -lh
```

Comparez les tailles des fichiers TPSYS.tar et TPSYS.tar.gz. Que constatez-vous ?

la compression a beaucoup réduit la taille du fichier compressé: TPSYS.tar.gz par rapport au fichier tpsys.tar

(Remarque: il fallait re-archiver pour comparer car le dossier archivé précédemment a été désarchiver)

```
[Saras-Mac:archive mbpro$ ls -lh
total 16
drwxr-xr-x  2 mbpro  staff    64B Jun  1 18:45 tpsys
-rw-r--r--  1 mbpro  staff   1.5K Jun  1 19:19 tpsys.tar
-rw-r--r--  1 mbpro  staff   116B Jun  1 18:59 tpsys.tar.gz
```

Écrire une seule commande qui permet de trouver tous les fichiers (.c) et de les compresser ensuite.

~ gzip *.c ~