***COMMANDES LINUX :***

* Fermer Terminal : **exit**
* Eteindre le PC : **poweroff**
* Afficher le chemin absolu du répertoire actuel : **pwd**
* Afficher msg : **echo** *ma\_chaine*
* Afficher la valeur d’une variable : **$var**
* Affecter une valeur à une variable : **var =** *18*
* Afficher Date : **date**
* Information system : **uname –a**
* Information session : **who am i**
* Afficher toutes les instructions utilisées : **history**
* Entrer dans un répertoire : **cd** *mon\_répertoire*
* Lister le contenu du répertoire en cour : **ls**

Avec détails : **ls –l**

* lister le cntenu d’un répertoire à distance : **ls ../***rep*
* Sortir du répertoire courant : **cd ..**
* Affichage page par page : **ls –l | more**
* Exécution d’un fichier bash : **bash** *nom\_fichier.sh*
* Générer l’exécutable d’un fichier C :  
   1er méthode : **gcc** *fichier.c* **–o**  *fichier*

2ème méthode : **gcc –c** *fichier.c* **gcc –o** *fichier fichier.c*

* Exécuter un fichier C : **./***fichier*
* Nom utilisateur : **logname**
* Information matériel ordinateur : **id**
* Info numéro terminal : **tty**
* Afficher le contenu d’un fichier : **cat** *fichier1 fichier2*
* Concaténer plusieurs fichier : **cat** *fichier1 fichier2*
* Afficher nbr de lignes, mots et octets d’un fichier : **wc** *fichier*
* Afficher date et heur des connections a linux : **who**
* Afficher l’espace disque : **df**
* Lister les variables d’environnements : **env**
* Afficher les numéros des lignes : **cat -n**

***Manipuler fichier & répertoire :***

* Création d’un répertoire : **mkdir** *rep*
* Création d’un fichier :

1er méthode : **gedit** *fichier*

2ème méthode : **cat>** *fichier (CTRL+D pour finir)*

3ème méthode : **touch** *fichier1 fichier2*

* Déplacer un fichier : **mv** *fichier1 fichier2 rep/fichier*
* Copier fichier :

1er cas , fichier vide : **cp** *rep/fichier rep/destination*

2ème cas ,fichier non vide : : **cp –r** *rep/fichier rep/destination*

* Supprimer fichier : **unlink** *fichier*
* Supprimer répertoire :

1er cas , répertoire vide : **rmdir** *rep/rep*

2ème cas , répertoire non vide : **rm –r** *rep/rep*

* Afficher les droits d’accès : **ls –l**  *rep/fichier*

***DROITS D’ACCES :***

* Modifier les droits d’accès :

Tout modifier : **chmod** *u=r rep ou fichier* ***chmod*** *666 rep ou fichier*

Révoquer : **chmod** u–r , g-w rep ou fichier

Ajouter : **chmod** *u + wx rep ou fichier*

*(u => user , g => group , o => others , a => all)*

*( r => read(4) , w => write (2) , x => execute (1) )*

* Lorsqu’on en ne possédant que les droits d’écriture sur un fichier,

il n’est pas possible de :

* L’ouvrir.
* Copier.

Il est possible de :

* D’écrire dedans avec la commande cat.
* Supprimer.
* Déplacer.
* Masque de création: **umask**

*(666 – le nbr = droits d’accès par défaut pour fichier)*

*(777 – le nbr = droits d’accès par défaut pour rep)*

* *Changer les droits d’accès par défaut :* ***umask***  026

***Redirection & archivage & compression :***

* Rediriger un msgd’erreur*: commande* ***2>*** *fichier*
* Rediriger un résultat : *commande* ***1>*** *fichier*
* Rediriger l’erreur et le résultat*: commande* ***2>****fichier****>>****fichier*
* trier selon le premier attribut :  **ls –l | sort**
* Rediriger vers un terminal : *commande* ***> /dev/pts/****0*
* Rediriger vers nul-part*: commande* ***>null***
* Création d’un lien physique : **ln** fichierSource fichierPhy
* Création d’un lien symbolique : **ln –s** *fichierSource fichierSym*
* Afficher le numéro d’inode (case mémoire pointée) : **ls –i** *fichierSource fichierSym fichierPhy*
* Afficher selon composition du nom du fichier :

Finissant par mot : **ls \****mot*

Commençant par mot : **ls** *mot***\***

Contenant mot : **ls \*** *mot* **\***

avec 1 lettre en plus : **ls** *mot{, ?}*

ne possédant aucun chiffre :  **ls** *mot****\*[ !0-9]***

avec 1 lettre ou 1 autre :  **[Aa]**

* Archivage en *.tar :* **tar cvf** *rep.tar rep*
* Désarchivage d’un *.tar :* **tar xvf** *rep.tar*
* Compression d’une archive *.tar en .tar.gz :* **gzip** *rep.tar   
   ou bien* **tar czf** *rep.tar.gz rep*
* Compression de tous les fichiers .c  : **tar czf** *fichierC.tar.gz* **\*.c**

ou bien **: mrdir** fichierC **| mv \*.c** fichierC **| tar cvf** fichierC.tarfichierC**|gzip** fichierC.tar

* Décompression : **gunzip** *rep.tar.gz*

***Lien dynamique & statique & library :***

* Edition de lien dynamique*:* **gcc –o** *fichier.dyn fichier.c*

*(le lien dynamique n’inclut pas les bibliothèque il leurs fait appel* ⬄ *petit volume de data)*

* Afficher bibliothèque requise pour exe un fichier dyn : **ldd** *fichier.dyn*
* Edition de lien statique : **gcc –static** *fichier.c fichier.stat*

*(le lien statique inclut les bibliothèques requises dans le fihier* ⬄ *grand volume de data)*

* Afficher le code objet (désassemblé) d’un fichier : **Objdump –d** *fichier.dyn fichier.o  
  (longueur code du fichier dynamique > longueur code du fichier objet)*
* Création d’une librairie statique : **ar r** *library.a fichier1.o fichier2.o*
* Edition de lien avec bibliothèque statique : **gcc –o** *fichierExe fichier.o library.a*puis  : **./***fichierExe*
* Création de bibliothèque dynamique : **gcc –shared -o** *library.so fichier1.o fichier2.o*
* Edition de lien avec bibliothèque dynamique : **gcc –o** *fichierExe fichier.o library.so  
  puis pour* ***.*/***fichierExe* 🡺 *il faut d’abord : copier la bibliothèque dans le même répertoire   
  et utiliser la commande* ***idconfig n*** rep/rep

***PID & PPID & premier et arrière plan :***

* Affichage du PID et PPID des processus : **ps –all**
* Suspendre l’exécution d’un processus : **CTRL + Z**
* Arrêter l’exécution d’un processus : **CTRL + C**
* Afficher l’état d’un processus : **ps -o stat** *PID*
* Remettre en premier plan un processus : **fg %***PID*
* Mettre en arrière plan unprocessus*:* **bg %***PID*

*Aussi :*

* *CTRL + Z ⬄ killtstp PID*
* *CTRL + C ⬄ killint PID*
* Exécuter un programme en arrière plan :

1ère méthode : **bg ./***fichier  
 2ème* méthode : **& ./***fichier*

* Affichage de la liste des processus en arrière plan : **jobs**
* Affichage de la hiérarchie des processus : **pstree –h** *PID*ou bien juste : **pstree –h**
* Affichage des infos sur les ressources utilisées par les processus : **top**
* Faire survivre un processus à la déconnexion du user (fermeture du terminal) : **nohup bash** fichier.sh

* Exécuter des commandes shell dans un programme C*:* **system(**"la\_commande"**)**
* Création d’un processus enfant : **fork()**

*‘Elle renvoie -1 en cas d’erreur, 0 si le processus exécuté et le processus fils, le PID du fils si le processus exécuté et le père)*

* Désactiver un signal : **trap ‘’** *num\_signal*
* Réactiver un signal : **trap** *num\_signal*
* Envoyer un signal : **kill –***num\_signal PID*

*(SIGKILL ne peut être dérouté)*

* Afficher la liste des signaux : **kill -l**

***PROGRAMMME :***

|  |
| --- |
| **int** main()  { **int** return\_value; return\_value = **system(**"ls -l /"**)**;  **return** return\_value;  } |

***Explication:***

* Exécution de commande LINUX dans un fichier C.

|  |
| --- |
| #include <stdio.h>  #include <unistd.h>  **int** main()  **{** printf("L'identifiant du processus est %d\n", getpid()); printf("L'identifiant du processus parent est %d\n", getppid());  **return** 0;  } |

***Explication:***

* Affichage du PID et PPID du programme lui-même.

programme modulaire

1. Créer un fichier *add.c* contenant :

#includ "add.h" int add(int x,int y){return (x+y);}

* La commande *gedit add.c*

1. Créer un fichier *add.h* contenant :

Int add(int x,int y) ;

* La commande *gedit add.h*

1. Créer un fichier *test.c*:

#include <stdio.h>

#include "add.h"

int main() {

int a = 3, b=4,c;

c=add(a,b);

printf("%d + %d = %d\n",a,b,c);

return 0;

}

***Explication :***

* La commande *gcc –c test.c add.c*
* La commande *gcc –o test test.o add.o*
* La commande *./test*

*#include<stdio.h>*

*#include<unistd.h>*

*#include<stdlib.h>*

*#inclide<sys/wait.h>*

*Int main()*

*{*

*pid\_t pid, stat;*

*int status;*

*printf("Père %d: processus fils non encore créé\n", getpid());*

*pid = fork();*

*if (pid != ­1){*

*if (pid != 0){*

*printf("Père %d : Création du fils %d\n", getpid(), pid);*

*wait(­&status);*

*if(WIFEXITED(status))*

*{*

*printf("Code de retour du fils : %d\n",*

*WEXITSTATUS(status));*

*}*

*}*

*else {*

*printf("Fils %d : je suis créé par %d\n",getpid(),getppid());*

*execlp( "ls ", "ls ", "-l ",NULL) ;*

*exit(5);*

*}*

*}*

*else*

*printf("Père %d: Création impossible du processus fils\n");*

*return 0;*

*}*

***Explication :***

* Le programme affiche d'abord un message sans condition
* «Père *PPID*: Processus fils non encore créé».
* Création d’un processus avec la commande : *fork()*
* et envoi le résultat dans la variable *PID*.
* Tester la valeur du *PID*:
  + Si *PID* <> ­-1:

Signifie que la création a réussie.

* + Si *PID<>-1 && PID <>0*,

signifie que c'est le père qui exécute l'instruction

(Le *fork()*renvoie le *PID* du fils créé au père)

«Père *PPID* : Création du fils *PID*».

* + Si *PID = 0*

Signifie que le fils exécute l'instruction

(Le *fork()*renvoie la valeur 0 au fils créé)

« Fils *PID* : je suis créé par *PPID*».

* + Si *PID = -1*

La création a échoué

«Création impossible du processus fils».

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

**int** nsig[NSIG];

**void** handler(**int** s)

**{** printf("Signal %d recu %d fois\n", s, ++nsig[s]); **}**

**int** main(**void**)

**{ int** s; **for** (s = 1; s < NSIG; s++)

**{ if** (signal(s, handler) == SIG\_ERR)

fprintf(stdout, "Le signal no %d ne peut être dérouté\n", s);

nsig[s] = 0; **} while**(1) pause();

}

***Explication :***

* Se programme déroute les signaux dé routable  
  en calculant le nombre de fois ou il est lancé sans l'exécuter

|  |
| --- |
| #!/bin/bash  i=0  **while** true  **do**  i=`expr $i + 1`  echo "Valeur de i : $i" >> hello  sleep 2  **done** |

***Explication :***

* Fichier **bash**, qui envoi la chaine ne caractère dans un fichier *hello.*