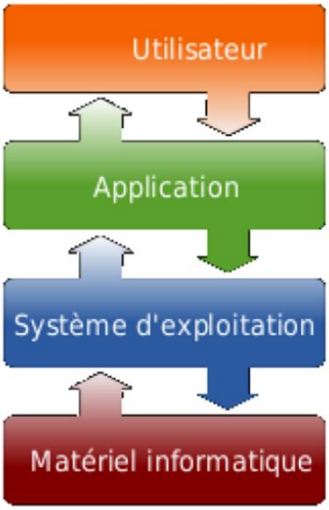
Système d'exploitation (Operating System libre Linux)

1. Introduction

- Le système d'exploitation est le logiciel le plus important pour la machine, c'est lui qui :
  - gère les ressources : processeurs, mémoires disques,
  - périphériques …
  - fournit une base pour le développement et l'exécution
  - des programmes d'application.
- Il a pour but :
  - D'offrir une vue simple, uniforme et cohérente de la
  - machine.
  - De protéger le système et ses usagers de fausses
  - manipulations.
  - de faciliter la programmation des applications.
- L'OS est l'intermédiaire entre les logiciels applicatifs et le matériel



  - Un programme d'application ne peut pas accéder directement au matériel, il doit passer par l'OS

- Du point de vue des utilisateurs il doit fournir:
  - Une vue uniforme des entrées/sorties
  - Une mémoire virtuelle et partageable
  - La gestion de fichiers et répertoires
  - La gestion de droits d'accès, sécurité et du traitement
  - des erreurs
  - La gestion de processus
  - La gestion des communication inter-processus
- Du point de vue des ressources, il doit assurer :
  - le bon fonctionnement des ressources et le respect des délais
  - Identification de l'usager d'une ressource
  - Le contrôle des accès et les interruptions aux ressources
  - La gestion des erreurs
  - L'évitement des conflits

2. Caractéristiques principales d'un OS

- Gestion des processus
  - Programme qui s'exécute avec l'ensemble des données et information nécessaires à son exécution
  -
  - L'OS doit : créer, gérer, synchroniser et faire communiquer les processus.
- Gestion de la mémoire
  - Stockage des données et instructions en cours d'exécution (mémoire centrale)

- L'OS doit allouer la mémoire disponible aux processus et optimiser son utilisation.

➢ Gestion des entrées/sorties
- Communication avec les périphériques externes
- L'OS doit assurer la communication avec les périphériques quelques soit les particularités matérielles.

➢ Gestion des fichiers
- Stockage et organisation des données dans la mémoire secondaire.
- L'OS doit gérer la création, la suppression et l'accès aux fichiers avec les sécurités nécessaires.

➢ Gestion du fenêtrage
- Partage d'un écran en plusieurs fenêtres constituant autant d'IHM pour les processus.
- L'OS Gère le fenêtrage.

➢ Gestion des réseaux
- Communication entre machine.
- L'OS doit : permettre d'établir des liaisons pour le partage des ressources et l'échange d'informations.

➢ Gestion de systèmes répartis
- Partage des ressources matérielles et des données voir des programmes
- L'OS doit fournir une architecture client/serveur

3. Les types d'OS

➢ **Multipostes** : plusieurs utilisateurs peuvent partager les ressources de la machine.

➢ **Multitâches** : plusieurs processus peuvent s'exécuter à la fois.

➢ **Multi-processeurs** : répartitions des processus sur plusieurs processeurs.

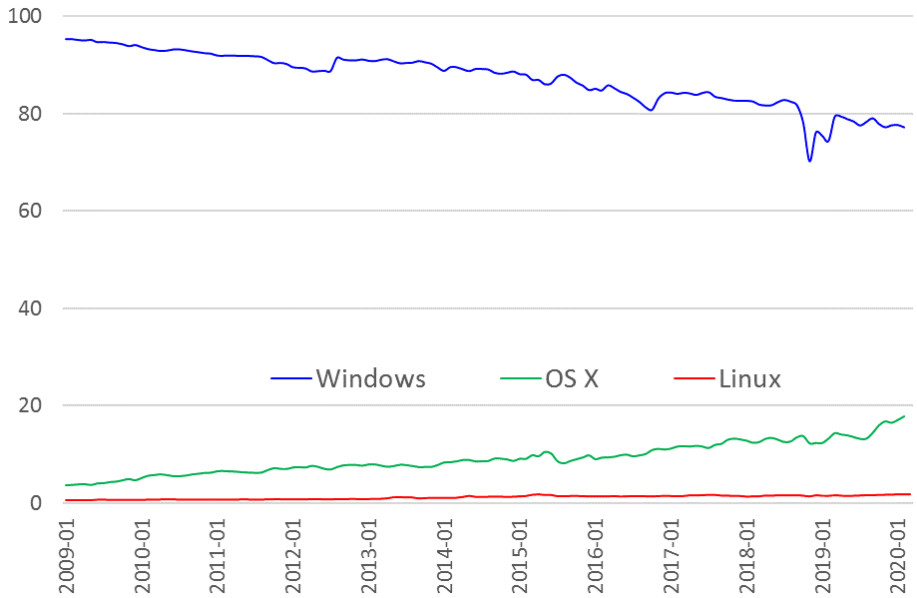
➢ **Temps reel : RTOS** (**R**eal **T**ime **O**perating **S**ystem) garantit que les opérations seront effectuées en respectant des délais précis.

Système d'exploitation (Operating System libre Linux)

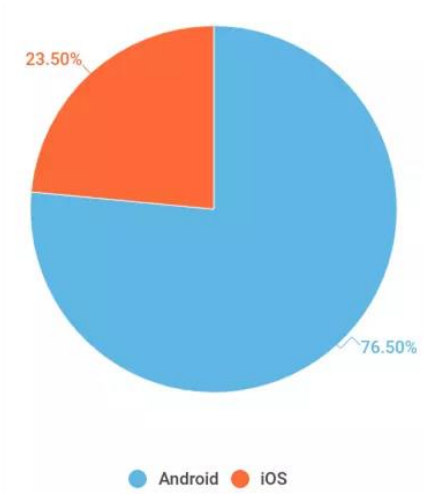
4. Les différents OS

Apple, Inc.	Précurseurs	Logiciel Système 5 · Système 6 · Système 7 · NeXTSTEP
	Mac OS	Mac OS 8 · Mac OS 9 · Darwin · Mac OS X · Mac OS X Server
Dérivés de BeOS		BlueEyedOS · Inferno · Haiku · ZETA
IBM		AIX · MVS · OS/2 · OS/360 · OS/390 · OS/400
Microsoft Windows	MS-DOS et dérivés	MS-DOS · 3.x · 95 · 98 · Me
	Branche NT	NT · 2000 · XP · 2003 · Vista · 2008 · 7
POSIX / UNIX	BSD	FreeBSD · NetBSD · OpenBSD · DragonFly BSD · PC-BSD
	GNU-Linux (Liste)	Debian · Fedora · Gentoo · Mandriva · Red Hat · Slackware · SuSE · Ubuntu
	Autres dérivés	HP-UX · LynxOS · Minix · QNX · Solaris · System V
Antédiluviens		AmigaOS · Plan 9 · QDOS · TOS · VMS
Autres systèmes		eyeOS · FreeDOS · MenuetOS · ReactOS · VxWorks
Formes particulières	Embarquée	Cisco IOS · iPhone OS · Palm OS · Palm webOS · Symbian OS · Windows CE
	LiveCD / LiveUSB	Knoppix · Slax

PC bureau



Mobile France fin 2019



Système d'exploitation (Operating System libre Linux)

5. Les Systèmes de fichiers

- Structure de données permettant de **stocker** et d'**organiser** des informations dans des fichiers sur des **mémoires secondaires** (disque dur, disquette, CD-ROM, clés USB, disques SSD, etc.).
- Les fichiers sont organisés dans une **arborescence de répertoire.**
- Différentes méthodes permettent d'associer un nom de fichier à son contenu.
  - **EXT3 :** (Unix) les fichiers et les répertoires sont identifiés par un numéro unique **(inode).** Ce numéro permet d'accéder à une structure de données regroupant toutes les informations sur un fichier (protection, date, …) (<4Tio)
  - **Linux Swap**, utilisé par le système Linux. Ce système sert à gérer le fichier d'échange de Linux.
  - **FAT32 :** (Windows) chaque répertoire contient une table associant les noms de fichier à leur taille et un index pointant vers la table d'allocation de fichiers. Une zone réservée du disque indique pour chaque bloc de données l'index du bloc suivant du même fichier. (<2Tio)
  - **NTFS :** (Windows) fonctionnant un peu à la façon d'une base de données.
  - Par rapport à FAT, il gère les droits (ACL) sur les fichiers et répertoires , le chiffrement des fichiers (EFS) la compression des fichiers , les quotas par volume. (< 256Tio)
  - exFAT (Extended File Allocation Table) : conçu tout particulièrement pour les mémoires flash, support des AC

Système d'exploitation	Types de système de fichiers supportés
Dos	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows NT4	FAT, NTFS (version 4)
Windows 2000/XP	FAT, FAT16, FAT32, NTFS (versions 4 et 5)
Linux	Ext2, Ext3, ReiserFS, Linux Swap(, FAT16, FAT32, NTFS)
MacOS	HFS (Hierarchical File System), MFS (Macintosh File System)
OS/2	HPFS (High Performance File System)
SGI IRIX	XFS
FreeBSD, OpenBSD	UFS (Unix File System)
Sun Solaris	UFS (Unix File System)

Système d'exploitation (Operating System libre Linux)

6. La sécurité

- **Windows :** cible préférée des pirates. Le problème ne vient pas du fait qu'il est moins sûr que les autres systèmes mais plutôt de sa popularité.
- **Mac OS X :** système qui est censé être particulièrement sûr, mais pas sans failles.
- **Linux :** grande réactivité pour combler les failles.
- **Android :** le système étant ouvert et récent il est très prisé des personnes malveillantes, son utilisation sur des terminaux connectés ultra-portables demande des précautions particulières à l'utilisateur.

7. Les listes de contrôle d'accès ACL

- **Désigne deux choses en sécurité informatique :**
  - un système permettant de faire une gestion plus fine des droits d'accès aux fichiers que ne le permet la méthode employée par les systèmes UNIX.
  - en réseau, une liste des adresses et ports autorisés ou interdits par un pare-feu.
  - Sous UNIX, les ACL ne remplacent pas la méthode habituelle des droits. Elle s'ajoutent à elle au sein de la norme POSIX
- Les ACL sont implémentées par le système de fichiers NTFS
- Mac OS X gère les ACL depuis la version 10.4

(Librement inspiré de https://debian-facile.org)

8. Les commandes les plus utilisées

8.1. Les bases

Pour certaines tâches, l'utilisation de la ligne de commande avec un shell s'avère bien plus pratique et plus puissante que la souris en mode graphique.

- La ligne de commande dans le terminal ou dans la console.

Pour écrire en toute sécurité vos lignes de commande.

- Auto-complétion

Visualiser l'ensemble des dernières commandes que vous avez saisies dans votre console.

- history

Le manuel : pour tout savoir sur une commande

- man

Comprendre les droits d'accès aux fichiers.

- Protection des fichiers

8.2. La syntaxe d'une commande

- Une commande se présente souvent de cette manière :

```
commande [option(s)] argument(s)
```

exemple :

```
ls -l /home/monRépertoire
```

- `ls` est la commande permettant d'afficher les fichiers contenus dans un répertoire.
- `-l` est une des options de la commande `ls` qui lui spécifie d'afficher beaucoup plus d'informations sur chacun des fichiers.
- `/home/monRépertoire` est l'argument qui indique le répertoire dont le contenu doit être affiché.

Un répertoire est un fichier spécial pouvant contenir des fichiers … qui peuvent être des fichiers de type répertoire … etc.

- Options des commandes

Pour connaître les options de chaque commande, vous pouvez faire précéder votre commande de la mention la commande man ou la commande info.

```
man ls
info ls
```

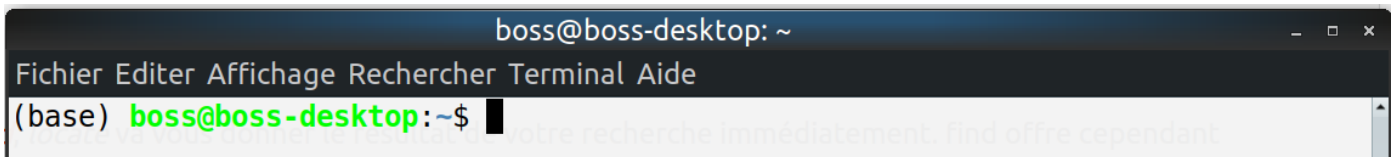
Taper `Q` pour quitter la documentation.

8.3. Les premières commandes linux essentielles

8.3.1. Ouverture du terminal

- **CTRL+ALT+T**
- Recherche du terminal dans la distribution par exemple mais ça peut être autrement : Cliquer sur menu puis terminal ou encore

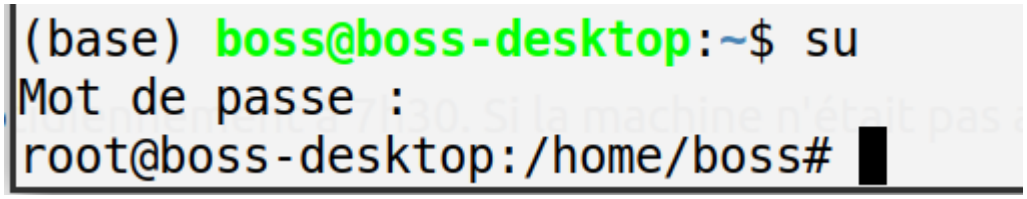
On obtient une fenêtre de commande / de console où on peut intéragir avec le **"shell".** Un shell Unix est un **interpréteur de commandes** destiné aux systèmes d'exploitation Unix et de type Unix qui permet d'**accéder aux fonctionnalités internes du système d'exploitation**. Il se présente sous la forme d'une in**terface en ligne de commande** accessible depuis **la console** ou **un terminal**.



**boss@boss-desktop:~$** ici l'utilisateur s'appelle boss la machine boss-desktop et :~$ signifie que c'est un utilisateur standard.

Les différents types d'utilisateurs :

- **Standard :** possède les droits sur les commande normales et ses fichiers
- **L'administrateur root** : possède tous les droits
  - Procédure pour configurer le mot de passe root
    - boss@boss-desktop:~$ sudo passwd root
    - [sudo] Mot de passe de boss :
    - Entrez le nouveau mot de passe UNIX :
    - Retapez le nouveau mot de passe UNIX :
    - passwd : le mot de passe a été mis à jour avec succès
  - Accès à une séssion root



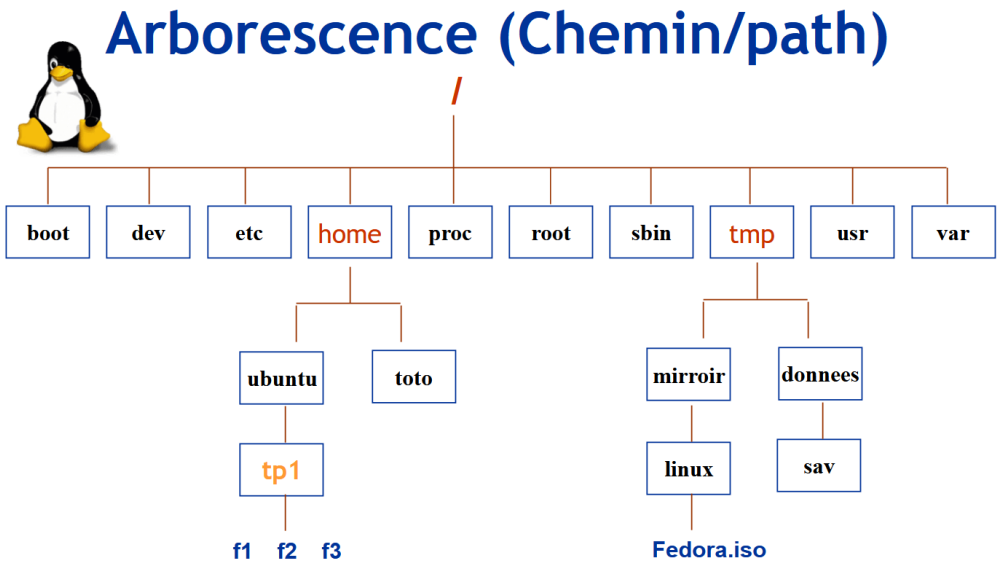
Le nom de l'utilisateur se change en root et le prompt devient un #

De manière générale il est déconseillé de se loguer en root pour la sécurité.

On préfére se faire faire passer pour le super utilisateur grace à la commande sudo (pour *Super User DO*) qui vous permet d'obtenir temporairement les privilèges de superutilisateur pour effectuer des tâches ponctuelles requérant ce niveau pour s'exécuter.

```
(base) boss@boss-desktop:~$ sudo apt upgrade
[sudo] Mot de passe de boss :
Lecture des listes de paquets... Fait
```

8.3.2. les répertoires sous linux



Répertoire courant : /home
Nom répertoire de destination : tp1

Chemin absolu : /home/ubuntu/tp1

Chemin relatif : ubuntu/tp1

Répertoire courant : /tmp
Nom fichier de destination : Fedora.iso

Chemin absolu : /tmp/mirroir/linux/Fedora.iso

Chemin relatif : mirroir/linux/Fedora.iso

Répertoire	Description du répertoire	intérêt d'un système de fichiers dédié

/	"racine", point d'entrée du système de fichiers	oui (obligatoire)
/boot	noyau Linux et l'amorceur	non (sauf exception)
/bin	exécutables de base, comme par exemple cp, mv, ls, etc...	non
/dev	fichiers spéciaux nommés devices qui permettent le lien avec les périphériques de la machine	oui (automatique)
/etc	fichiers de configuration du système	oui (délicat)
/home	fichiers personnels des utilisateurs (un sous-répertoire par utilisateur)	oui
/lib	librairies et les modules du noyau (/lib/modules)	non
/media	« points de montage » des médias usuels : cd, dvd, disquette, clef usb	non
/root	personnel de l'administrateur	non
/sbin	exécutables destinés à l'administration du système	non
/tmp	fichiers temporaires	oui
/usr	exécutables des programmes (/usr/bin et /usr/sbin), la documentation (/usr/doc), et les programmes pour le serveur graphique (/usr/X11R6).	non (en général)
/var	fichiers qui servent à la maintenance du système (les fichiers de journaux notamment dans /var/log)	oui

8.3.3. les commandes pour parcourir l'arborecence répertoires

- Où suis-je positionné dans le système, dans quel répertoire j'erre...?

  pwd : *"print working directory" pwd*

```
(base) boss@boss-desktop:~$ pwd
/home/boss
```

- Lister les fichiers d'un répertoire
  - ls

```
(base) boss@boss-desktop:~$ ls
anaconda3              essai.py    Musique      Vidéos
```

- Créer un répertoire
  - mkdir

```
(base) boss@boss-desktop:~$ mkdir Essai
(base) boss@boss-desktop:~$ ls
anaconda3              Essai       Modèles    pyzo-4.8.1       test_qt.py
```

- Supprimer un répertoire vide
  - rmdir

```
(base) boss@boss-desktop:~$ rmdir Essai
(base) boss@boss-desktop:~$ ls
anaconda3                essai.py  Musique  Qt                Vidéos
```

- Se déplacer dans les répertoires
  - cd

```
(base) boss@boss-desktop:~$ cd Vidéos
(base) boss@boss-desktop:~/Vidéos$
```

pour remonter au répertoire parent cd ..

```
(base) boss@boss-desktop:~/Vidéos$ cd ..
(base) boss@boss-desktop:~$
```

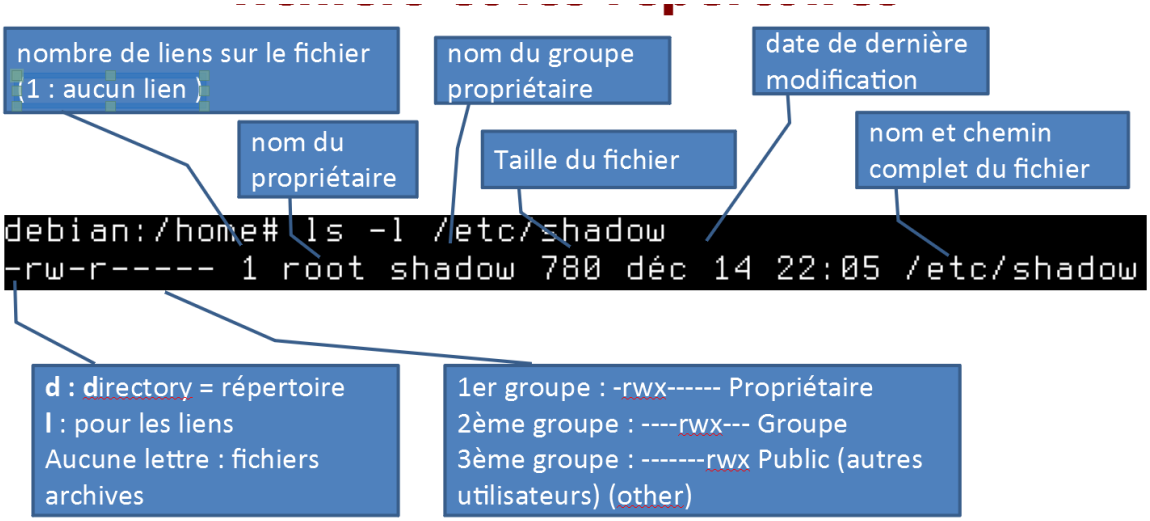
le répertoire lui même :"."

```
(base) boss@boss-desktop:~$ cd .
(base) boss@boss-desktop:~$
```

revenir dans **son propre répertoire cd ~**

8.3.4. Les fichiers

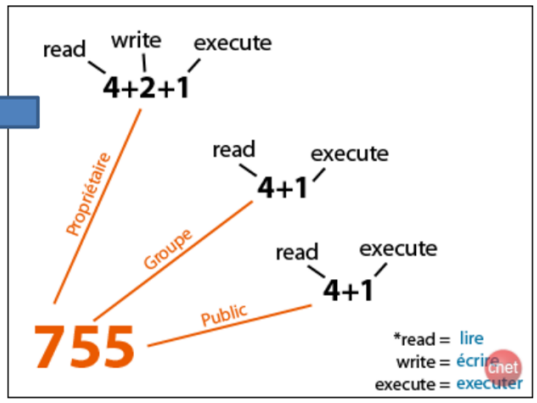
- Les droits des fichiers



Symbole		Sur un fichier normal	Sur un répertoire
r	read (lire)	lire, copier le contenu	visibilité des fichiers directement contenus
w	write (écrire)	modifier le contenu	création ET destruction des fichiers directement contenus
x	execute (executer)	exécution	entrée et parcours du rép.
-	absence de droit		

- chmod

Chmod 764 text.txt



- **u** : utilisateur,
- g : groupes
- o (other) :autres utilisateurs
- a (all) :t out le monde (u,g,o)

- \+ droits ajoutés
- \- droits supprimés
- = droit égal à

- r pour read
- w pour write
- x comme execution

chmod –c a=r, gu+w, u+x text.txt
chmod -v u=rwx, g=rw, o=r text.t

- Retrouver un fichier

find

- Voir le contenu d'un fichier
  - cat
  - more
  - less
  - most
- Trouver du texte dans un fichier
  - grep
- Éditer un fichier
  - nano
- Créer un fichier
  - > (Le chevron)

```
(base) boss@boss-desktop:~$ > mon_texte.txt
(base) boss@boss-desktop:~$ ls
anaconda3             essai.py    mon_texte.txt  pyzo-4.8.1       test_qt.py
```

  - touch

```
(base) boss@boss-desktop:~$ touch mon_texte2.txt
(base) boss@boss-desktop:~$ ls
anaconda3             essai.py    mon_texte2.txt  Public       Téléchargements
```

- Copier un fichier
  - cp

- Déplacer ou renommer un fichier
    - mv
- Supprimer un fichier
    - rm
- Archivage de données
    - tar, rar

8.3.5. Administration et gestion du système.

- L'espace disque
    - L'espace occupé
        - du
    - Les espaces occupés et disponibles
        - df
- La gestion des processus
    - top
    - ps
    - kill
    - htop

9. Rédaction de scripts Shell

#! et exécution

La première ligne d'un script shell doit toujours commencer par #!, suivi ensuite de l'interpréteur de commande à utiliser. Si vous n'utilisez pas de commande propres à bash ou zsh, vous pouvez laisser l'interpréteur de commande par défaut :

```
#!/bin/sh
```

Si vous voulez pouvoir exécuter votre script, n'oubliez pas de donner les droits correspondant à l'utilisateur devant l'exécuter.

Par exemple :

```
chmod a+x monscript.sh
```

Ou encore :

```
chmod root:admin monscript.sh
chmod 750 monscript.sh
```

ex récupération des arguments

L'argument 0 est le chemin utilisé pour exécuter le programme.

Exemple, cela peut-être ./monscript ou monscript s'il est dans le PATH, ou même ./un-lien-vers-mon-script si vous avec utilisé ln. Il est accessible via la variable $0.

Le premier argument est ensuite accessible via $1, le deuxième via $2, etc. L'ensemble des arguments est accessible via $@ alors que la concaténation de tous les arguments (séparés par des espaces) est accessible via $*.Le nombre d'arguments est accessible via $#.
Voir : variables-de-substitution-predefinies-principalement-dans-les-scripts

La commande shift permet de décaler tous les arguments vers la gauche ($1 désigne le second, etc.).

Exemple :

```
#!/bin/sh

echo "Commande initiale : $0 $@"

N=0
while [ -n "$1" ];
do
  N=$(($N+1))
  echo "Argument $N : $1";
  shift;
done
```

execution

Pour lancer le script en ligne de commande écrire $ ./nom_du_programme.sh

10. Pipe

Un *pipe* permet à deux processus de s'échanger des données. Les commandes Shell permettent de créer facilement un *pipe* :

```
envoyeur | receveur
```

Dans cet exemple, la sortie standard de **envoyeur** est reliée à l'entrée standard de **receveur**.
Le Shell est un outil très pratique pour créer un *pipe*, mais en aucun cas les données du *pipe* ne transiteront par le Shell lors de l'exécution.
Un *pipe* est aussi appelé FIFO (First In First Out) cela signifie que les données qui sortiront en premier de **envoyeur** seront également les données qui rentreront en premier dans **receveur**.

Cependant, l'ordre est celui dans lequel les appels-système à WRITE et READ ont été invoqués, pas l'ordre dans lequel les processus tentent de les réaliser.

Par défaut, les *pipe* sont bloquants, ce qui signifie que lorsque **envoyeur** écrit dans le *pipe*, il s'endort jusqu'à ce que le processus **receveur** ait terminé de TOUT lire, ce qui réveille **envoyeur**.
Si **receveur** est prêt à lire dans le *pipe* mais qu'**envoyeur** n'y a encore rien écrit, **receveur** s'endort et se réveillera dès qu'il y aura quelque chose à lire.

Lorsque vous programmez une application, vous pouvez choisir de rendre les *pipes* non-bloquants, grâce au flag O_NDELAY ou O_NONBLOCK.
Un *pipe* peut très bien être bloquant d'un côté et non-bloquant de l'autre côté.
L'ordre de réception des données peut différer suivant que le *pipe* soit bloquant (par défaut) ou non-bloquant (flag O_NDELAY).

Le caractère pipe est « | » et s'obtient avec la combinaison de touches : `Alt Gr` avec le `6` du clavier des **lettres**.

Les redirections d'entrée/sortie[2)] permettent de rediriger les résultats vers un fichier.
Ce fichier peut ensuite être réinjecté dans un filtre pour en extraire d'autres résultats.
Cela oblige à taper deux lignes :

1. une pour la redirection vers un fichier,
2. l'autre pour rediriger ce fichier vers le filtre.

Les tubes ou pipes permettent de rediriger directement le canal de sortie → d'une commande vers → le canal d'entrée d'autre.

Ainsi, les 2 redirections suivantes :

`ls -l > resultat.txt` Puis : `wc < resultat.txt`

Deviennent cette commande unique :

`ls -l | wc`

11. Activité

- Créer un répertoire repertoire_linux
- Créer un fichier essai_1.txt grâce au >
- Créer les fichiers test_1.txt, test_2.txt grâce à la commande touch
- Remplissez un fichier liste_fichier.txt avec le contenu du répertoire, puis visionner son contenu avec la commande cat.
- Grace à la commande grep lister les fichiers commençant par test , puis ranger le tout dans un fichier resultat.txt. On désire ensuite uniquement connaitre le nombre de fichiers en utilisant la commande wc. Utiliser une PIPE (tube) pour afficher ce résultat. Enfin écrivez un script shell pour le faire.
- Faites tourner le programme « récupération des arguments » en créant un script programme_exemple.sh

```
BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ > essai_1.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ ls
essai_1.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ touch test_1.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ touch test_2.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ ls
essai_1.txt  test_1.txt  test_2.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ ls > liste_fichier.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ cat liste_fichier.txt
essai_1.txt
liste_fichier.txt
test_1.txt
test_2.txt
```

```
BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ grep test liste_fichier.txt
test_1.txt
test_2.txt
```

```
BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ grep test liste_fichier.txt  | wc -l
2
```

```
BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ grep test liste_fichier.txt  > resultat.txt

BOSS@GILLES-TRAVAIL MINGW64 /repertoire_linux
$ ls
essai_1.txt  liste_fichier.txt  resultat.txt  resultats.txt  test_1.txt  test_2.txt
```