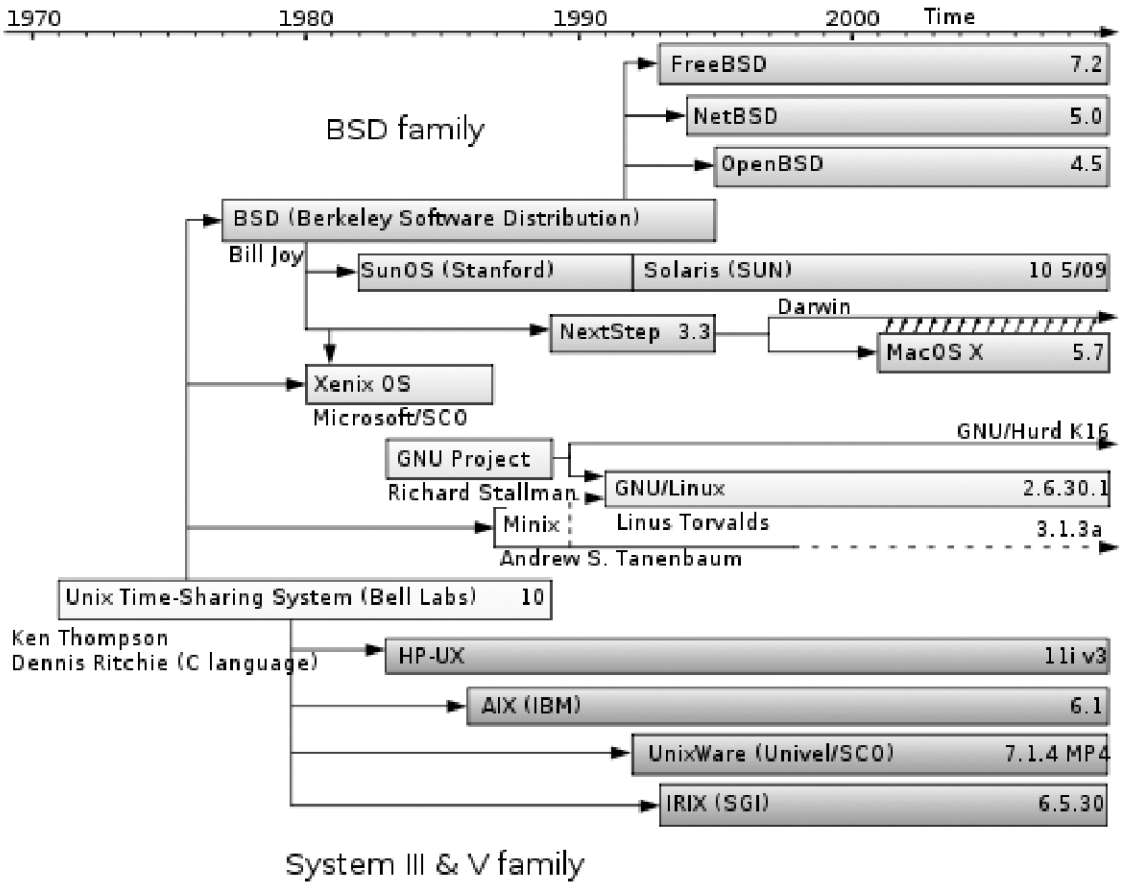
1) HISTORIQUE ET VERSIONS



2) GRANDS PRINCIPES

Basé sur un noyau (kernel) : cœur du système d'exploitation

Système d'exploitation multitâches : (sous)programmes ou processus gérant "en parallèle" processus, mémoire, fichiers, périphériques, réseau,… :

Matériel (hardware) <=> UNIX (OS) <=> Applications (software)

Système d'exploitation multi-utilisateurs : plusieurs utilisateurs sur le même système central via des terminaux

Système de fichiers : sous UNIX tout est fichier (3 types de fichiers : ordinaires txt / bin, répertoires, spécifiques périphériques)

Le Shell : interpréteur de commandes & langage scripts. Interface entre l'utilisateur et le système d'exploitation (terminal). Différentes versions avec spécificités : sh, bash, csh, ksh,…

Outils fournis : programmes & utilitaires pour utiliser l'OS (commandes : ls, mkdir, grep, ps, sh, cal, man, mail, vi, tar,…)

3) OUVERTURE ET FERMETURE DE SESSION

Ouverture d'une session :
Accéder au système avec un (ou des) compte(s) utilisateur(s)
Plusieurs sessions possibles (indépendantes et multitâches)

En local : UNIX + Applications installés et exécutés sur l'ordinateur utilisé (via interface graphique ou console texte)

Sur une machine distante (host) : UNIX + Applications installés et exécutés sur l'ordinateur distant (= le serveur). Accès par un terminal ou par une émulation de terminal (= le client, ex : telnet) via un réseau série / LAN / Internet (TCP/IP)

Exemple de connexion par telnet :
telnet 192.168.10.200
login : Ldupont
password : mypassword
$

Login : identifiant utilisateur connu du système pour se connecter (unique)

Password : mot de passe de connexion pour ce login

Invite ou prompt (Shell) :
Login => Connexion => Shell => Prompt d'attente de saisie d'une commande
Symbole : $ ou # ou > ou % ou autre

Syntaxe générale d'une commande : cde -opt(s) arg(s)

Commande : nom de fonction, script exécutable, programme,…
Option(s) : caractère(s) précédé(s) par '-' (cas de traitement)
Argument(s) : donnée(s) traitée(s) par la commande

Erreur si syntaxe non respectée (message). **Aide sur une commande :** « man cde »

Option(s) et Argument(s) ne sont pas toujours présents ou obligatoires. Ils dépendent de la commande et de son contexte d'utilisation.

Fermeture d'une session (Shell) : CTRL+D ou « exit » ou « logout »

Arrêt complet du système : uniquement par l'utilisateur « root » (super-utilisateur) en ligne de commande : « halt » ou « shutdown -h now » ou « init 0 ». Arrêt de tous les services et programmes + déconnexion de tous les utilisateurs (!)

A l'ouverture d'un Shell, l'utilisateur se trouve par défaut dans son répertoire de travail (ou dossier personnel) : /home/nom_utilisateur/

4) GESTION DES FICHIERS ET DES REPERTOIRES

Le système de fichiers : structure d'accueil des données sur un support physique (D7, HD, CD, DVD, USB,...)

Structure du système de fichiers Unix/linux : composé…

- **Des fichiers ordinaires :** suites d'octets txt / bin
- **Des fichiers spéciaux :** références à des périphériques
- **Des répertoires :** fichiers listes de noms avec identifiants inodes
- **D'une table d'inodes :** références à tous les fichiers + informations
- **D'un super bloc :** bloc de données en début de partition (informations sur le système de fichiers, taille de la table d'inodes, taille des blocs,…)

Arborescence des fichiers sous UNIX :
Organisation selon une structure arborescente
Racine = '/'
1 seule racine pour tout le système (≠ MS-DOS/Windows : 1 racine par lecteur)

Le FHS (Filesystem Hierarchy Standard) : norme d'uniformisation des emplacements et noms des fichiers + répertoires dans les arborescences UNIX

Voir les « inodes » des fichiers et des répertoires : ls -i
Lister les répertoires et fichiers à la racine : ls -la /
Lister toute l'arborescence système depuis la racine : ls -R /

- **/ :** Racine du système
- **/bin :** Exécutables des commandes essentielles
- **/boot :** Fichiers statiques du chargeur d'amorçage
- **/dev :** Fichiers spéciaux des périphériques
- **/etc :** Fichiers de configuration
- **/home :** Répertoires personnels des utilisateurs
- **/lib :** Bibliothèques partagées essentielles et modules du noyau
- **/media :** Contient les points de montages pour les médias amovibles
- **/mnt :** Point de montage pour monter temporairement un système de fichiers
- **/proc :** Répertoire virtuel pour les informations système (noyaux 2.4 et 2.6)
- **/root :** Répertoire personnel du super-utilisateur
- **/sbin :** Exécutables système essentiels

Noms des fichiers : lettres minuscules et majuscules (attention : casse respectée), chiffres et caractères spéciaux. '/' interdit, '-' interdit au début, '.' cache le fichier si au début

Chemin absolu : chemin complet depuis la racine '/'
Chemin relatif : chemin depuis le répertoire courant ('..' pour remonter au parent)

Lister le contenu d'un répertoire : ls dossier (avec chemin absolu ou relatif)
Options : -a (tous) ; -l (long) ; -i (inode) ; -R (récursif) ; -F (type)

Lister le contenu du répertoire courant : ls -la

Détail des informations affichées :

```
total 16
drwxr-xr-x    5 sebastie  sebastie   170 May  1 10:47  .
drwx------   46 sebastie  sebastie  1564 May  1 14:44  ..
-rw-r--r--    1 sebastie  sebastie  6148 May  1 10:47 .DS_Store
drwxr-xr-x    6 sebastie  sebastie   204 Apr 30 12:45  Administratif
drwxr-xr-x    3 sebastie  sebastie   102 May  1 19:44  BUX-SJ
```

=> Type du fichier : défini par un attribut (caractère de gauche) : '-' (fichier standard) ; 'd' (répertoire) ; 'l' (lien)

=> Autorisations d'accès au fichier : gérées par une combinaison de 3 x 3 caractères :
3 car. pour Propriétaire (u) / 3 car. pour Groupe (g) / 3 car. pour Autre (o)
Avec les droits par catégorie : - (pas le droit) / r (read) / w (write) / x (exec ou répertoire)

=> Nombre de fichiers pour les répertoires (sinon 1)
=> Nom du propriétaire et du groupe pour le fichier
=> Taille en octets du fichier
=> Date et heure de dernière modification sur le fichier
=> Nom du fichier ou du répertoire ('.' = caché)

Création d'un fichier (plusieurs possibilités) : résultat d'une commande (ls, cat), copie (cp), éditeur (vi, nano), nouveau fichier vide (touch),…

Création d'un nouveau fichier : cat > newfile (saisie & CTRL+D pour enregistrer)
Afficher son contenu : cat newfile

Suppression d'un fichier (!) : rm newfile

Copie, déplacement et renommage d'un fichier :

Copie : cp f1 f2 ; cp f1 /rep/f2 ; cp f1 /rep
Déplacement (dans un autre répertoire) : mv f1 /rep
Renommage (dans le même répertoire) : mv f1 f2

Création d'un répertoire : mkdir rep ; mkdir -p /rep/rep1/rep2 (avec parents)

Suppression d'un répertoire (!) : rmdir rep1 (si vide + droits) ; rm -r rep1 (si non vide)

Changement de répertoire : cd rep (avec chemin absolu ou relatif)

Affichage du chemin depuis la racine ('/') au répertoire courant : pwd

Les liens physiques et symboliques (links) : commande « ln »
Crée un lien (physique ou symbolique) vers un fichier ou un répertoire. Options :
aucune : crée un lien physique (sorte de copie liée au fichier d'origine)
-s : crée un lien symbolique (similaire au raccourci Windows)
Exemple :
ln -s /Chemin/MonFichier /AutreChemin/MonLien
Crée un lien symbolique MonLien de /Chemin/MonFichier dans le répertoire /AutreChemin/
(remarque : utiliser les chemins en absolu)

Sauvegarde, archivage et restauration :

Commande « tar » : sauvegarde une arborescence complète dans un fichier dédié et restaure ces mêmes fichiers (similaire au « zip » Windows). Syntaxes :
Archivage & compression : **tar -cvzf archive.tar rep/**
Décompression & désarchivage : **tar -xvzf archive.tar**

Commande « dd » : sauvegarde et restauration d'un disque complet, système de fichiers, fichier, blocs, zone disque,… revient à créer un fichier « iso » (image disque).

Modification des autorisations sur les fichiers (rwx) : possible que par le propriétaire du fichier ou l'administrateur (super-utilisateur « root » ou avec « sudo »)

Commande générale : chmod *code* fichier
Avec code numérique (en octal) : chmod 777 fichier ; chmod 000 fichier ; etc.
Avec code symbolique : chmod ugo+x fichier ; chmod ugo-x fichier ; etc.
(symboles : 'u', 'g', 'o', 'a', '+', '-', '=', 'r', 'w', 'x')

Droits d'accès par défaut aux fichiers et répertoires (en octal inversé) : umask *code*

Changement de propriétaire et de groupe sur un fichier : par l'administrateur

Propriétaire (owner) : sudo chown nouveau-proprietaire fichier
Groupe : sudo chgrp nouveau-groupe fichier

Autres commandes : mkfs (formatage de disques), mount / umount (rattachement / détachement d'un device dans l'arborescence), fsck (vérification d'un système de fichiers).

5) ETUDE DU SHELL

Le Shell :

Programme **interpréteur** de commandes et de langage scripts.
Interface entre l'utilisateur et le système d'exploitation (terminal virtuel ou physique).
Différentes versions avec spécificités : sh, bash, csh, ksh, psh,…
A l'utilisation (commandes et scripts), toujours connaître le Shell utilisé (risques d'incompatibilités).

Syntaxe générale d'une commande : cde -opt(s) arg(s)

Commande : nom de fonction, script exécutable, programme,…
Option(s) : caractère(s) précédé(s) par '-' (cas de traitement)
Argument(s) : donnée(s) traitée(s) par la commande

Erreur si syntaxe non respectée (message). **Aide sur une commande :** « man cde »

Cycle d'exécution d'une commande :

1) L'utilisateur saisit la commande (invite / prompt) + ENTREE
2) Le Shell vérifie la syntaxe et recherche la commande : d'abord dans « $PATH » puis dans le répertoire courant
3) Si OK, le programme est exécuté (durée variable)
4) La fin du programme retourne au Shell (invite / prompt)

Interruption de l'exécution d'une commande : CTRL+C (ou CTRL+D)

Les Shell disponibles & les commandes principales Unix/Linux se trouvent dans les répertoires système : /bin/ et /sbin/

Remarque : « $PATH » est une variable système qui contient un ensemble de répertoires (ordonnés) dans lesquels le Shell va d'abord rechercher les programmes/commandes à exécuter. Dès que le programme est trouvé dans un répertoire, celui-ci est exécuté.

Exécution successive : séparée par **';'** (exemple : **pwd ; ls**)

Regroupement de commandes : pour exécution en tâche de fond ou pour redirection de la sortie de plusieurs programmes.
Entre parenthèses () : exécution dans un nouveau processus Shell
Entre accolades {} : exécution dans le processus Shell courant
Exemples : (ps ; ls ; pwd ; id ;) > fichier.txt
{ ps ; ls ; pwd ; id ; } > fichier.txt

Exécution conditionnelle (très utilisée en script Shell) : commandes séparées par
'||' : la commande suivante s'exécute si la précédente est KO
'&&' : la commande suivante s'exécute si la précédente est OK

Suspendre et reprendre l'exécution d'une commande :
Suspendre un programme : **CTRL+Z => [n] : n° de job**
Ramener le programme en 1er plan : **fg %1 (si n° de job = [1])**
Tester avec : « sleep 3600 »

Gestion des tâches : processus / process = programme en cours ou en attente d'exécution en mémoire (multitâche), identifié par un numéro unique PID (16 bits) + informations en mémoire (table des processus).

Exécution en tâche de fond : exécution en arrière plan (background)
Faire suivre la commande **d'un espace et d'un '&' + ENTREE**
=> Affiche [n] : n° de job + PID : n° de processus (unique)
L'invite / prompt se réaffiche ensuite immédiatement (saisie d'autres commandes)
Exemple : « sleep 3600 & »

jobs : liste des programmes + n° jobs en tâche de fond ([n])
fg %n : ramène la tâche en premier plan (exécution foreground)
bg %n : ramène la tâche en arrière plan (exécution background)
ps : liste des processus avec leur PID (n° unique)
kill : tue un processus en précisant son PID (exemple : « kill -9 1234 »)

Quelques caractères spéciaux du Shell (jokers) :

. désigne le répertoire courant
.. désigne le répertoire parent
~ désigne le répertoire d'accueil (répertoire de travail de l'utilisateur connecté)
? remplace un caractère quelconque
***** remplace une chaîne de caractères quelconque
introduit un commentaire
$ introduit un nom de variable
& lance la commande en background
; sépare 2 commandes

Variables du Shell et variables d'environnement :

Zones mémoire réservées et nommées stockant des données. Perdues (vidées) quand la session du Shell est fermée/quittée.

Variables du Shell : utilisées que par le Shell courant.

Variables d'environnement (partagées) : transmises aux processus fils (uniquement), grâce à la commande « export nomvariable ».

Création d'une variable : valeur1=10 ; repertoire='/tmp' ; tabchaine=(chaine1 chaine2 chaine3)

Visualisation des variables définies : set (toutes) ; env (que environnement/exportées)

Evaluer une variable : obtenir son contenu (précédée de '$'). Exemples :
echo $valeur1 ; echo $repertoire ; echo $PATH

Modifier le contenu d'une variable :
Réaffecter : valeur1=12
Compléter : repertoire=$repertoire/script => /tmp/script
Exemple : PATH=$PATH:/home/utilisateur

Suppression de variable : unset valeur1 repertoire tabchaine

Variable en lecture seule : readonly valeur1

Autres opérations sur les variables :

Nombre de caractères des données : **nbcar=${#var}**
Extraction d'une partie du contenu : **res=${var:index:nbre}**
Affectation de données à un tableau : **tab=(elt1 elt2 …) ; tab=([1]=elt1 [2]=elt2 …) ; tab[1]=elt1 ; tab[2]=elt2 ; …**
Evaluation des éléments d'un tableau : **${tab[n]} ; ${tab[*]}**

Variables système (echo $...) : HOME, PATH, PS1, PS2, PWD,…

Variables spéciales (echo $...) : ?, 0 à 9, 10 et +, #, $, *, @

Le code retour des commandes : echo $?
(code = 0 : commande OK / code <> 0 : commande KO)

Les arguments (variables) passés aux commandes / scripts Shell : $1 $2 $3 …

Modifier l'invite / prompt : PS1='# '

Fonctions du Shell :

Sous-programmes nommés en mémoire, contenant des commandes et mots clés Unix.
Perdus quand la session du Shell est fermée/quittée.

Création d'une fonction : en ligne de commande ou dans un script. Exemple :

```
nomfonction()
{
  ls
  cd $HOME
}
```

Exécution d'une fonction : nomfonction + ENTREE

Visualisation des fonctions du Shell : set ; typeset -f ; declare -f

Rendre une fonction exportable (environnement/partagée) : transmise aux processus fils (uniquement).

Export : export -f nomfonction
Fin d'export : export -nf nomfonction
Fonctions d'environnement : env

Supprimer une fonction : unset -f nomfonction

Fonction en lecture seule : readonly -f nomfonction

Exemple de fonction avec arguments :

```
calculer() { expr $1 $2 $3; }
```

(« expr » est une commande Unix de calcul : expr 3 + 5)

La substitution de commande : permet de récupérer le résultat d'une commande dans une variable ou pour une autre commande.

**Syntaxe (` obtenu par AltGr+7) :** « variable=`commande` » ou « variable=$(commande) »

Exemples :

```
aujourdhui=`date` ; echo $aujourdhui
compteur=0
compteur=`expr $compteur + 1`
echo $compteur
cd $(echo $HOME)
```

6) REDIRECTIONS ET FILTRES

Redirection en sortie :

Au lieu d'afficher le résultat d'une commande à l'écran (sortie standard), le **rediriger dans un fichier** à l'aide des symboles '>' (fichier créé ou écrasé) ou '>>' (fichier créé ou complété).

Syntaxe : « commande1 > fichier » et « commande2 >> fichier »
Exemples :
ls -la > fichier1
set > fichier1
cat > fichier2
cat >> fichier2

Redirection en entrée :

Au lieu d'utiliser le résultat d'une commande saisie au clavier (entrée standard), le **récupérer depuis un fichier** à l'aide du symbole '<' (redirection temporaire du clavier : '<<').

Syntaxe : « commande < fichier »
Exemples :
wc < fichier1
grep TERM < fichier1
Combinaison entrée/sortie :
wc < fichier1 > fichier2
cat > fichier3 << FIN

Filtre :

Renvoyer la sortie (le résultat) d'une 1ère commande vers l'entrée d'une 2ème commande, à l'aide du **symbole '|' (pipe obtenu par AltGr+6)**. **La 2ème commande exploite le résultat de la 1ère** pour en filtrer les données.

Syntaxe : « commande1 | commande2 »
Quelques commandes orientées filtres : more, sort, grep, wc, cut, tr, head, tail,…
Exemples :
ls -la | more
ls | wc
set | sort
set | grep TERM
set | sort | more
cat fichier | sort | more

Rechercher du texte dans un fichier : « grep -options chaîne fichier »

La commande « grep » permet de **rechercher une chaîne de caractères dans un fichier** (éventuellement à l'aide des expressions régulières). Options :

-i : ignore la casse (minuscules/majuscules)
-v : affiche les lignes ne contenant pas la chaîne
-c : compte le nombre de lignes contenant la chaîne
-n : chaque ligne contenant la chaîne est numérotée
-x : ligne correspondant exactement à la chaîne
-w : que les mots entiers

Exemple avec le fichier « carnet-adresses » contenant les lignes :
olivier:29:0298333242:Brest
marcel:13:0466342233:Gardagnes
myriam:30:0434214452:Nimes
yvonne:92:013344433:Palaiseau

Avec la commande : **grep Brest carnet-adresses**
Permet d'obtenir toutes les lignes contenant la chaîne de caractères 'Brest', soit :
« olivier:29:0298333242:Brest »

Rechercher des fichiers dans l'arborescence (à partir d'un répertoire) :

find . -name 'rep*'
sudo find / -name passwd

7) EDITEUR DE TEXTE VI

1er éditeur de texte plein écran, présent dans tous les systèmes Unix/Linux.

Accès en ligne de commande : « vi fichier »
Fonctionnement (3 modes) :
Commande : mode insertion (touche [i]), déplacement, correction, copier,…
Insertion : mode saisie du texte (buffer)
Exécution (caractère ':') : commandes enregistrer, quitter,…

Editer un texte avec « vi » (touche [Echap] si bloqué), cas simple :

vi texte.txt
[i]
saisie du texte
[Echap]
:wq

8) GESTION DES UTILISATEURS

Principe général :

Sur les systèmes d'exploitation Unix/Linux, l'utilisation du système passe par l'identification (login avec mot de passe) d'**un utilisateur (unique) appartenant à un groupe (unique)** ayant certains droits sur les fichiers. Tous les fichiers ont des droits définis ('rwx' + propriétaire + groupe). Cette organisation sécurise les systèmes Unix.

root : le super-utilisateur ayant tous les droits (administrateur)
sudo : mot-clé permettant d'exécuter une commande avec les droits « root »

Gestion des groupes et des utilisateurs :

Par 3 fichiers système définis selon la hiérarchie ci-dessous. Pour les visualiser :
« cat /etc/group » « cat /etc/passwd » « sudo cat /etc/shadow »

1) Le fichier texte /etc/group : chaque ligne = 1 groupe unique. **4 champs** (séparés par ':') : nom en clair, champ vide (x), identifiant sur 16 bits (GID), liste des utilisateurs membres.

2) Le fichier texte /etc/passwd : chaque ligne = 1 utilisateur unique. **7 champs** (séparés par ':') : nom de connexion, champ vide (x), identifiant sur 16 bits (UID), groupe (GID), informations optionnelles de l'utilisateur, répertoire d'accueil, programme Shell de login.

3) Le fichier texte /etc/shadow : mots de passe cryptés par utilisateurs. **9 champs** (séparés par ':') : nom d'utilisateur, mot de passe crypté (via « passwd »), jour de création (> 01/01/1970), jours avant changement possible, jours de changement obligatoire, autres infos (non utilisées).

Procédure simple de création d'un groupe avec un utilisateur :

1) Créer le groupe d'utilisateurs nommé « groupetest » : **sudo groupadd groupetest**

2) Afficher le contenu du fichier des groupes d'utilisateurs : **cat /etc/group** **(=> GID)**

3) Créer l'utilisateur nommé « usertest » appartenant au groupe « groupetest » (si GID = 1111) : **sudo adduser --gid 1111 usertest**

4) Afficher le contenu des fichiers utilisateurs (UID et mot de passe) : **cat /etc/passwd**
sudo cat /etc/shadow

5) Changer de mot de passe de « usertest » : **sudo passwd usertest**

6) Se connecter avec « usertest » : **su usertest** (quitter sa session : « exit »)

Recommandation pratique : toujours créer les groupes avant les utilisateurs.

9) SCRIPTS SHELL

Programmes Shell **écrits en fichier texte** (ASCII).
Avec fonctions, commandes, mots clés, #,…
Une commande script peut être un autre script Shell ou un programme exécutable.
1ère ligne : #!/bin/bash (pour utiliser le Shell « bash »).

Création d'un script : avec analyse préalable du besoin + éditeur de texte ('vi' ou autres)

Structure d'un script : linéaire et séquentielle (instructions exécutées en série)

Exécution d'un script : interprétation (pas de compilation)

Rendre le script exécutable : **chmod ugo+x monscript.sh**
Exécution en fonction du Shell utilisé (interpréteur) :
« . monscript.sh » (Shell courant) ou « bash monscript.sh »
(si répertoire courant ou $PATH, sinon préciser le chemin relatif ou absolu)

Suppression d'un script : rm monscript.sh

Structure « if » : instructions conditionnelles « si, alors, sinon ». Mots clés : if, then, else, fi. Imbrications possibles (if dans if).

Structure simple :

```
if expression
  then
    instructions
fi
```

Structure avec « else » (sinon) :

```
if expression
  then
    instructions
  else
    autres instructions
fi
```

Structure « while » : boucles / itérations « tant que, faire ». Mots clés : while, do, done. Imbrications possibles.

Structure générale :

```
while expression
  do
    instructions
  done
```

Syntaxe des expressions pour « if » et « while » : « [valeur1 opérateur valeur2] »
Valeurs : variables ($), nombres (123), chaîne de caractères ("abc"), un caractère ('a')
Opérateurs de comparaison avec une chaîne de caractères : =, !=, >, <,...
Opérateurs de comparaison avec un nombre : -eq, -ne, -gt, -lt, -ge, -le
Opérateurs de comparaison logique : -a (and), -o (or), ! (not)

La commande « clear » : efface l'écran

La commande « read » : lit une information

Saisir au clavier des informations (texte, nombre) et les stocker dans une variable :
read variable
read -p "Saisissez votre nom : " nom
read -p "Saisissez votre âge : " age
Point d'attente dans un programme (ENTREE pour continuer) : **read**

La commande « echo » : affiche une information

Afficher à l'écran un texte ou un nombre :
echo "Bonjour"
echo 3.14
Afficher à l'écran l'information (texte, nombre) d'une variable :
echo $variable
echo "Votre nom est : " $nom
echo "Votre âge est : " $age
Saut de ligne à l'écran : **echo**

1er Exemple :
```
#!/bin/bash
clear
read -p "Veuillez saisir votre nom : " nom
if [ $nom = "MARTIN" ]
  then
    echo "OK"
  else
    echo "KO"
fi
```

2ème Exemple :
```
#!/bin/bash
compteur=1
while [ $compteur -le 1000 ]
  do
    clear
    echo $compteur
    compteur=`expr $compteur + 1`
  done
```