1 Objectifs de ce document

Ce document a pour objectif de poursuivre l'utilisation des systèmes d'exploitation initié en première. Il vous permettra d'identifier l'architecture de linux. Dans une deuxième partie, vous comprendrez la gestion des processus et des ressources par un système d'exploitation et du risque de l'interblocage (deadlock). Pour finir, vous verrez deux architectures qui se développent énormément pour les objets connectés, la domotique, la voiture connectée : les micro-contrôleurs et les systèmes sur puce (SoC).

Rappel : si vous n'avez pas de linux installé, vous pouvez utiliser les commandes du shell dans le bash proposé sur le site pythonanywhere. (vu en TP)

Ce document se base :

— sur le cours Jonathan Lejeune, Maitre de Conférences à Sorbonne Université (ex-UPMC) et intégré au Laboratoire d'Informatique de Paris 6 (LIP6) dans l'équipe Inria Delys.

— sur le livre Linux : programmation et communication de J-M Rifflet et de J-B Yunès, Dunod 2003 et notamment les chapitres 1, 2, 3 et 4.

— sur le livre Tle Nsi, T.Balabonski - S.Conchon - JC Filliâtre - K Nguyen, Ellipses

— Cours Cnam DIU

2 Quelques mots sur les systèmes d'exploitation :

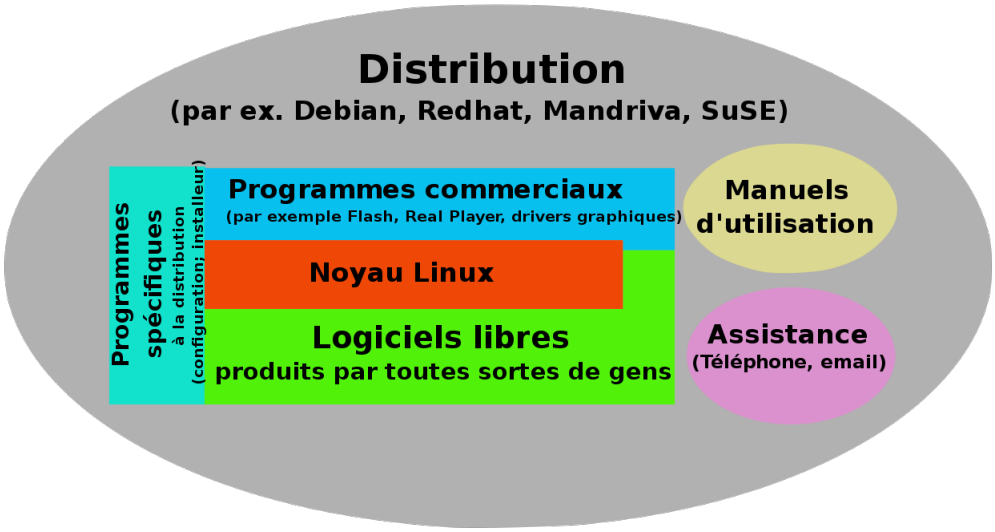
Le système d'exploitation est une couche logicielle permettant de faire l'interface entre le matériel et les applications des utilisateurs de la machine. Il est notamment responsable de la gestion des ressources de la machine comme le processeur, la mémoire et les périphériques comme les disques dur, les cartes réseaux (Wifi, Ethernet), la carte graphique, la carte son, etc...

Linux est un des premiers systèmes d'exploitation (OS) apparu dans les années 70 et fut le système d'exploitation le plus utilisé jusqu'à la fin des années 80. Le développement d'une multitude de versions propriétaires pour différentes architectures et l'émergence au début des années 90 de Windows, plutôt adapté aux utilisateurs non-informaticiens, ont causé une énorme perte de popularité de Linux.

De nos jours Linux désigne surtout une famille de systèmes d'exploitation plutôt qu'un OS à part entière. On note donc deux familles de systèmes d'exploitation : Windows et Linux. Parmi les OS Linux qui ont résisté à la popularité de Windows, on peut citer BSD, Linux ou encore MacOS X qui reprennent de plus en plus d'ampleur dans le grand public.

Une distribution Linux est une collection de logiciels pour la plupart open-source composée du système d'exploitation Linux (on parle aussi de noyau Linux) et d'une suite de librairies, d'utilitaires ou applications. Chaque distribution est maintenue par une communauté de développeurs permettant ainsi de produire des mises à jour régulières des différents logiciels associés à la distribution. En revanche, le noyau Linux est développé indépendamment de toute distribution. Il constitue donc un point commun entre elles. Parmi les distributions existantes, on note des distributions mères (ArchLinux, Debian, Gentoo, RedHat, Slackware, Android) qui se différencient par exemple par les outils utilitaires de gestion de paquets (aptitude pour Debian, Rpm pour RedHat, ...). De ces distributions mères ont été développées des distributions filles qui fournissent les logiciels de leur distribution mère associée et d'autres utilitaires ou applications qui leur sont propres.

Depuis 1991, GNU/Linux (GNU is Not Linux : Linus Torvalds et Richards Stallman sont les créateurs de ce logiciel libre) est un système d'exploitation fiable, fortement utilisé : ordinateurs, téléphones, box Internet, objets connectés, 23 des 25 sites les plus visités du monde utilisent des serveurs Linux et la totalité des supercalculateurs. Linux représente à lui seul entre 75% à 80% de tous les serveurs Web dans le monde.



Le gros avantage du noyau Linux indépendant est qu'on peut le recompiler, l'adapter à ses usages car c'est un logiciel dont on donne le code source, en plus d'être gratuit.

3 Utilisation du Terminal :

L'utilisation et l'administration d'un ordinateur peuvent se faire de manière graphique grâce à des fenêtres et à l'utilisation d'une souris (comme le font la plupart des utilisateurs) ou bien en mode texte à travers une console ou un terminal où les commandes sont directement données au clavier. Lorsqu'on lance l'application "Terminal" , le système ouvre une fenêtre puis exécute un programme (ou processus) particulier appelé interpréteur de commande ou shell reconnaissable par la présence d'une chaîne de caractères appelée invite ou prompt. Ceci invite l'utilisateur à entrer des commandes de son choix. Le prompt par défaut se termine par un caractère comme $, % ou # précédé par une chaîne de caractères tel que le login de l'utilisateur et le nom de la machine séparés par un .

Lorsqu'on ouvre un terminal : on a une invite au debut avec "nomutilisateur @ nom de la machine", exemple sur la raspberry pi :

pi@192.168.43.123 :~ $

L'invocation d'une commande consiste à écrire le nom de cette commande, suivi d'un ou plusieurs autres mots appelés arguments ou paramètres et de taper ensuite sur la touche Entrée :

$nom\_commande\ [argument_1 ... argument_n]$

Les différentes chaînes de caractères constituant le nom et les arguments d'une commande sont séparées par au moins un caractère d'espacement. Les arguments d'une commande peuvent être :

- des paramètres optionnels ou options permettant d'obtenir des variantes de comportement de la commande : ils sont, le plus souvent, précédés du caractère - ou de la chaîne - -.
- des arguments ordinaires sur lesquels la commande doit être appliquée (par exemple des noms de fichiers, d'utilisateurs, un nombre, etc.) Après lecture d'une ligne, l'interpréteur shell l'analyse, l'exécute et ré-affiche le prompt une fois que la commande se

termine pour permettre à l'utilisateur de saisir la commande suivante. Lorsque le shell détecte une erreur dans la commande (par exemple : commande introuvable ou erreur de paramètre(s)) il affiche un message d'erreur.

4 Quelques commandes élémentaires :

1. - Man est ton ami : **la commande man**

   Sous Linux il existe la commande d'aide man qui permet d'afficher sur le terminal la notice d'utilisation d'une commande. Pour plus d'information sur cette commande, tapez :

   man man

   Pour sortir d'une page d'un manuel, tapez la lettre q. (comme quit).

2. - **La commande echo**

   La commande echo permet d'afficher sur la sortie standard chacun de ses arguments. Par exemple, pour afficher Bonjour tout le monde sur le terminal, il faut taper la commande suivante :

   echo "Bonjour tout le monde"

   ou alors

   echo Bonjour tout le monde

   On pourra remarquer que la commande echo ajoute implicitement un caractère de retour chariot à la fin (ce qui fait que le prompt de retour s'affiche toujours à la ligne suivante). Pour éviter de passer à la ligne il faut renseigner l'option -n :

   echo -n "Bonjour tout le monde"

3. - **La commande who**

   La commande who permet principalement de lister les utilisateurs qui sont actuellement connectés à la machine. Il est également possible d'afficher des informations complémentaires sur le système avec l'option -a

4. - **La commande pwd**

   Elle permet d'afficher le chemin d'accès vers le répertoire où se situe l'utilisateur qui a entré la commande.

5. - **La commande stat**

   En tapant la commande stat avec un nom de fichier comme argument, il est possible d'avoir des informations sur ce fichier maintenues par le système notamment :

   - les deux informations identifiant un fichier : l'i-noeud et le numéro de volume.
   - sa taille réelle en octet
   - le nombre de liens physiques
   - le type du fichier (fichier régulier, répertoire, lien symbolique, fichier spécial, etc.)
   - les dates d'accès, de modification et de création.

6. - **La commande expr**

   La commande expr permet d'évaluer des expressions sur des entiers. Il est également possible de faire des opérations sur les chaînes de caractères. En ce qui concerne les opérations entières, 3 arguments sont attendus :

   - le premier argument correspond à la première opérande
   - le deuxième argument correspond à l'opérateur
   - le troisième argument correspond à la deuxième opérande

   L'opérateur peut être :

   - + pour l'addition
   - - pour la soustraction
   - * pour la multiplication. Attention le caractère * étant un méta-caractère du shell il est nécessaire de protéger ce caractère avec des simples quotes (') ou bien en le précédent d'un n si on souhaite que le shell l'interprète comme un caractère classique.

- / pour la division entière
- % pour le reste de la division entière
- < ou <= pour tester si la valeur de opérande 1 est inférieure ou inférieure ou égale à celle de l'opérande 2
- > ou >= pour tester si la valeur de l'opérande 1 est supérieure ou supérieure ou égale à la valeur de l'opérande 2
- = pour tester si valeur de l'opérande 1 est égale à celle de l'opérande 2
- != pour tester si valeur de l'opérande 1 est différente de celle de l'opérande 2

7. - Expressions numériques

La commande expr permet d'évaluer des expressions arithmétiques. Cependant pour exploiter le résultat de l'expression, il est souvent utile de le stocker dans une variable. Pour cela il faut donc entourer la commande expr par des back-quotes ou bien par $(...). Afin de simplifier cette syntaxe, le shell offre un raccourci syntaxique en utilisant $((expression)). Cette syntaxe offre l'avantage également de rendre facultative l'utilisation des dollars lors de la lecture des variable. Si on considère l'existence de deux variables x et y, les 4 expressions suivantes sont équivalentes :

```
var='expr $x + $y'
var=$(expr $x + $y)
var=$(($x + $y))
var=$((x+y))
```

5 Les fichiers

1. - Fichiers et système de fichiers

Classiquement, un fichier représente un ensemble de données que l'on stocke de manière persistante sur un volume stable (disque dur, clé USB, ...). Ceci permet aux utilisateurs de retrouver leurs données même après extinction de l'ordinateur. Un fichier s'inscrit dans un système de fichiers. Le système de fichier définit :

- d'un point de vue logique une organisation hiérarchique des fichiers formant ainsi une arborescence. Cette organisation logique sera propre au système d'exploitation : par exemple sous Linux il n'existe qu'une seule racine impliquant une unique arborescence pour tous les volumes alors que sous Windows, ou pourra remarquer que chaque volume est représenté par une lettre (C :, D : ...) impliquant ainsi une arborescence par volume.
- d'un point de vue physique la façon dont les données sont organisées physiquement sur le volume. Cette organisation physique peut inclure des mécanismes de journalisation (pour gérer les erreurs par exemple) et diffère selon le système de fichiers choisi lors du formatage du volume. Par exemple le système de fichiers le plus répondu est NTFS pour Windows et Ext4 pour Linux.

2. - Comparaison entre fichiers et mémoire RAM

Contrairement aux données stockées dans un fichier, les données stockées dans la mémoire RAM sont (à l'heure actuelle) non persistantes, c'est à dire qu'elles sont perdues dès l'extinction de la machine qu'elle soit voulue ou non. En revanche le temps d'accès d'une donnée sur la RAM est considérablement plus petit qu'un accès disque (de l'ordre d'un facteur 1000). De plus la granularité d'accès est beaucoup plus fine sur la mémoire RAM que sur un disque. En effet l'accès aux données sur un disque se fait par bloc de données contigu physiquement (dont la taille est définie lors du formatage) alors que la RAM permet d'accéder directement à un octet donné (un octet est identifié par une adresse). Par conséquent, vouloir accéder à une donnée particulière sur un fichier nécessite de charger dans un premier temps le bloc contenant cette donnée en RAM et dans un second temps à la donnée voulue. Lire ou écrire dans un fichier nécessitent donc des mécanismes beaucoup plus complexes que les accès RAM. Il faut également noter qu'il est impossible d'allouer partiellement un bloc disque, ainsi la taille réelle d'un fichier (c'est à dire le nombre de données en octets que le fichier contient) est différente de sa taille physique qui est forcément un multiple de la taille d'un bloc.

3. - Les fichiers sous Linux

La notion de fichier sous Linux ne se résout pas uniquement au fichier disque classique. Il permet également de désigner une ressource physique ou logique de l'ordinateur (appelé device), comme par exemple un terminal, une imprimante, un disque physique ou logique (=partition du disque physique), la mémoire vive du système, un bus, une carte réseau, etc. Un fichier est donc caractérisé par son type qui définit l'ensemble des opérations que l'on peut lui appliquer. On peut classer les types de fichiers en deux grandes catégories :

- fichiers disques comme les fichiers réguliers, les répertoires, les liens symboliques ou les tubes nommés.
- fichiers spéciaux qui désignent une ressource du système

1. - L'arborescence de fichiers sous Linux

Les fichiers réguliers permettent de stoker des codes de programmes (binaires ou sources) ou bien des données classiques. Les répertoires sont des fichiers disques particuliers car ils permettent d'une part d'organiser le système de fichiers en arborescence et d'autre part de permettre de désigner de manière unique un lien vers un fichier du système. Un lien est un nom de fichier. Sous Linux, on retrouve bien souvent l'arborescence de la figure 3.1.

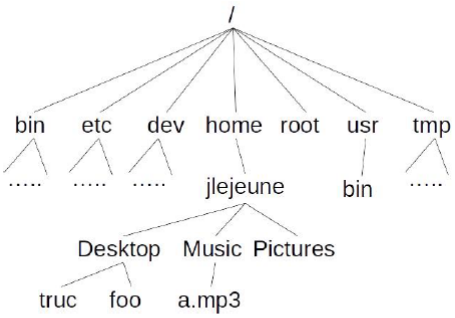


FIGURE 3.1 – Arborescence d'un système de fichiers UNIX

Le dossier le plus haut dans l'arborescence est le dossier racine et se désigne par /.

Un nom de fichier (ou lien) se désigne sans ambiguïté grâce au chemin qu'il faut parcourir depuis la racine du système pour l'atteindre, c'est le chemin absolu. Par exemple le lien foo se désigne par le chemin absolu suivant :

/home/jlejeune/Desktop/foo

Il est remarquable qu'il existe deux liens qui se nomment bin. Cependant, ce sont deux liens différents, l'un se désigne par /bin et l'autre par /usr/bin.

Dans un terminal, le shell est associé à un répertoire courant. Par défaut le répertoire courant du shell est la racine de votre répertoire personnel qui se désigne par un tilde $\sim$.

La commande pwd affiche le chemin absolu du répertoire courant. La notation de chemin absolu peut devenir très lourde notamment lorsque le lien à atteindre est très profond dans l'arborescence. C'est pourquoi il est possible de désigner des liens de manière relative (chemin relatif) au répertoire courant du shell. Par exemple, si le répertoire courant est /home/jlejeune alors il est possible de désigner le lien foo par le chemin relatif Desktop/foo.

2. - Les répertoires principaux dans les systèmes Linux

À titre informatif, voici les principaux répertoires que l'on retrouve dans les systèmes Linux :

- /bin et /usr/bin : contient les codes des commandes de bases du système.
- /etc : contient une sous-arborescence des fichiers de configuration système
- /dev : contient la sous-arborescence des fichiers spéciaux (les ressources)

- /home : contient les différentes arborescences des répertoires contenant les données des utilisateurs

- /root : répertoire du super-utilisateur dont le login est root. C'est l'utilisateur qui dispose de tous les droits pour les activités d'administration du système.

- /usr : contient une sous-arborescence où sont installés les programmes utilisateurs (plus ou moins l'équivalent du répertoire C : nprogrammes sous Windows)

- /tmp : répertoire contenant des fichiers temporaires utilisés/créés par les applications. Le contenu de ce répertoire est bien souvent effacé au moment du démarrage du système.

3. - Connaître le contenu d'un répertoire

Pour lister les fichiers d'un répertoire il faut utiliser **la commande ls**. Ainsi :

- ls affiche les fichiers du répertoire courant

- ls / affiche les fichiers de la racine de l'arborescence

- ls ~ affiche les fichiers du dossier personnel de l'utilisateur courant

On peut noter les options suivantes :

- -a pour afficher tous les fichiers y compris les fichiers cachés (le nom d'un fichier caché commence par un point).

- -l pour afficher davantage d'informations sur les fichiers.

4. - Parcourir le système de fichier

On peut remarquer lorsque l'on tape la commande ls -a rep que deux fichiers aux noms particuliers se sont affichés . et .. Ils correspondent respectivement au répertoire rep et au répertoire parent de rep. Ainsi il est toujours possible de désigner le répertoire courant par . et le répertoire parent du répertoire courant par .. Il est possible de changer de répertoire courant grâce à **la commande cd** . En tapant :

- cd .. le nouveau répertoire courant deviendra le parent du répertoire courant actuel

- cd rep le nouveau répertoire courant sera rep

- cd ou cd ~ permet de prendre comme répertoire courant le dossier personnel de l'utilisateur

- cd / permet de revenir à la racine

5. - Création d'une arborescence de fichiers

Pour créer une arborescence, nous avons besoins de deux commandes :

- **la commande mkdir** qui permet de créer un répertoire vide.

- **la commande touch** qui permet de changer la date de modification d'un fichier à la date courante ou créer un fichier régulier si celui-ci n'existe pas. Pour créer dans le répertoire personnel l'arborescence de la figure 3.2 (les fichiers réguliers sont en gras et entourés). Il faut donc taper les commandes suivantes :

```
mkdir -p essai/B/foo/bar
mkdir -p essai/B/bar
touch essai/A
touch essai/bar
touch essai/B/foo/bar/a.txt
touch essai/B/foo/bar/foo
```

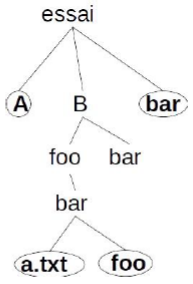


FIGURE 3.2 – Arborescence à créer

Pour information l'option -p de la commande mkdir permet de créer les répertoires intermédiaires (à savoir B et foo) si ces derniers n'existent pas. Il vous est possible de contrôler que l'arborescence correspond bien à celle attendu en tapant la commande ls -R qui permet d'afficher récursivement le contenu de tous les sous-répertoires. Vous avez la possibilité d'utiliser la commande tree si cette dernière est installée.

6. - Modification d'une arborescence de fichiers

Il est possible de modifier une arborescence :

- soit en créant un nouveau fichier à partir d'une copie d'un fichier déjà existant avec **la commande cp**.

- soit en déplaçant un fichier existant vers une autre branche de l'arborescence avec **la commande mv**.

- soit en supprimant un lien avec **la commande rm** ou bien **la commande rmdir** pour les dossiers vides.

On notera que déplacer un fichier est fondamentalement différent que de copier le fichier et de supprimer l'orignal ensuite. En effet, la copie implique la création d'un nouveau fichier (donc d'un nouveau i-noeud), de copier octet par octet les données du fichier source vers le fichier cible et ensuite de supprimer le lien du fichier original. Le déplacement est beaucoup plus simple (et donc moins coûteux en nombre d'opérations) car il nécessite juste de changer le nom du lien pour un même i-noeud à condition bien entendu que l'arborescence cible se trouve sur le même volume physique que l'emplacement original. La commande mv peut donc être vu comme une commande de renommage de fichier. Dans notre exemple on pourra :

- copier le fichier a.txt dans le répertoire essai/B/bar à l'aide de la commande

cp essai/B/foo/bar/a.txt essai/B/bar

- déplacer dans le répertoire essai/B/foo la copie du fichier a.txt que vous venez de mettre dans le répertoire essai/B/bar à l'aide de la commande

mv essai/B/bar/a.txt essai/B/foo/

- effacer le lien essai/B/foo/a.txt à l'aide de la commande

rm essai/B/foo/a.txt

- copier le dossier B/foo/bar dans le répertoire B/bar à l'aide de la commande

cp -r essai/B/foo/bar essai/B/bar/

On notera que l'option -r est obligatoire pour pouvoir copier récursivement la sous-arborescence d'un répertoire.

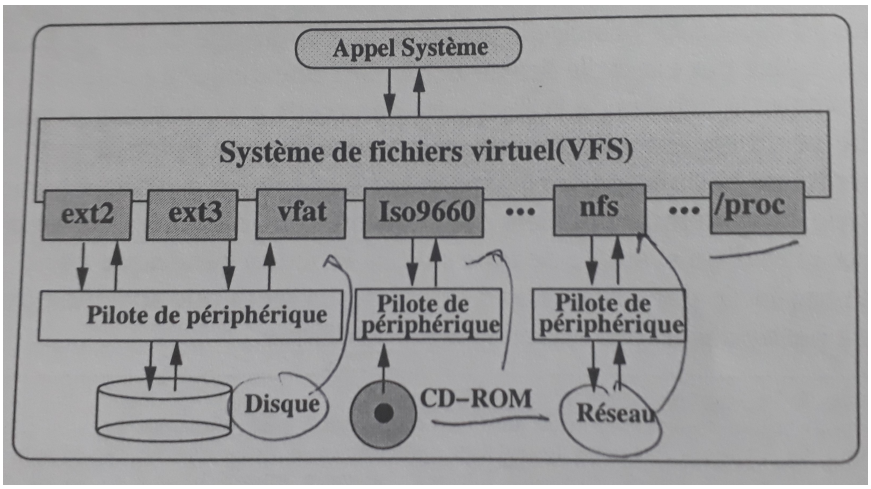
- effacer la copie du dossier que vous venez de faire l'aide de la commande

rm -r essai/B/bar/bar

Sur le même principe que cp, l'option -r est obligatoire pour pouvoir supprimer récursivement la sous-arborescence d'un répertoire.

Attention : On notera que la commande rm supprime définitivement un fichier (sauf si ce dernier possède un autre lien physique dans l'arborescence du système). En mode terminal il n'y a donc aucun répertoire intermédiaire type "corbeille" permettant de récupérer une éventuelle suppression par erreur. La commande rm est donc à manipuler avec prudence.

Important : la gestion des systèmes de fichiers (SGF) sous Linux est totalement différentes avec Windows. Le noyau de Linux va utiliser des appels Système avec un sytème de fichiers virtuel (VFS) :



Les appels de base seront par exemple :

— **mount/umount** : monter/démonter un système de fichiers

— **mkdir/rmdir** : créer/détruire des répertoires

— **open/close** : ouvrir/fermer des fichiers

— **read/write** : lire/écrire dans des fichiers

6 Les droits d'accès des fichiers Linux

Les différents fichiers (tout type confondu) sont associés à un utilisateur propriétaire. Afin d'éviter qu'un utilisateur n'accède à des fichiers dont il n'est pas propriétaire et sans autorisation, le système permet de définir des droits d'accès pour chaque fichier. Du point de vue d'un fichier donné, un utilisateur quelconque appartient à l'une des trois catégories suivantes :

- le propriétaire
- le groupe d'utilisateurs auquel le propriétaire appartient (le propriétaire exclu)
- les autres utilisateurs (le propriétaire et le groupe du propriétaire exclus)

On peut effectuer trois types d'opération sur un fichier :

- la lecture du contenu du fichier (identifié par le caractère r)
- l'écriture dans le fichier (identifié par le caractère w)
- l'exécution, c'est à dire pouvoir interpréter le fichier comme une commande (identifié par le caractère x)

Si le fichier est un répertoire :

- le droit en lecture signifie que l'on peut lister ses fichiers (commande ls)
- le droit en écriture signifie que l'on peut y créer ou supprimer des fichiers
- le droit en exécution signifie que l'on peut le prendre comme répertoire courant ou bien le traverser pour accéder à sa sous-arborescence.

Ainsi, pour un fichier donné il existe pour chaque catégorie d'utilisateur le droit ou le non-droit d'effectuer chaque type d'opération. Il y a donc 9 droits d'accès à positionner pour chaque fichier. Ces droits d'accès se représentent par trois triplets (un triplet représente les droits r, w et x) de bits valant 0 ou 1 indiquant si le droit en question est autorisé ou pas. Le premier triplet correspond aux droits du propriétaire, le second triplet correspond aux utilisateurs du groupe et le troisième triplet correspond aux droits des autres utilisateurs. Par exemple si un fichier a les droits d'accès suivant rwxr-xr– alors :

- le propriétaire possède les droits en lecture, écriture et exécution
- les utilisateurs appartenant au groupe du propriétaire possèdent les droits en lecture et exécution
- les autres utilisateurs possèdent uniquement le droit en lecture.

Pour afficher les droits des fichiers du répertoire courant, il est nécessaire de taper la commande ls -al.

Seul le propriétaire du fichier et le super-utilisateur peuvent changer ses droits d'accès. Pour changer les droits d'accès il faut utiliser la commande chmod. Il existe deux manières d'exprimer des modifications de droit

- Soit en octal où chaque triplet de bits est représenté par un chiffre entre 0 et 7 :

- $(0)_8 soit (000)_2 =$ – – – aucun droit
- $(1)_8 soit (001)_2 =$ – – x : exécution
- $(2)_8 soit (010)_2 =$ – w – : écriture
- $(3)_8 soit (011)_2 =$ –wx : écriture + exécution
- $(4)_8 soit (100)_2 =$ r – – : lecture
- $(5)_8 soit (101)_2 =$ r – x : lecture + exécution
- $(6)_8 soit (110)_2 =$ rw– : lecture + écriture
- $(7)_8 soit (111)_2 =$ rwx : tous les droits

Par exemple pour ajouter tous les droits à un fichier foo, il faut taper la commande :

chmod 777 foo

- soit avec un format prédéfini comportant 3 informations :

- le ou les triplets concerné(s) : a pour tous les triplets, u pour le triplet propriétaire, g pour le triplet groupe, et o pour les autres
- si il s'agit d'un ajout (caractère +) ou d'un retrait (caractère -) de droits
- les droits que l'on veut modifier : r pour la lecture, w pour l'écriture, x pour l'exécution

Par exemple pour ajouter les droits en lecture et en exécution au propriétaire et au groupe d'un fichier foo il faut taper la commande

chmod ug+rx foo

7 Le protocole ssh :

le protocole SSH permet de se connecter à une autre machine en utilisant son adresse IP. Le ssh est un service, un processus, il y a 65 536 ports : les 1024 premiers ports sont réservés aux protocoles : HTTP est sur le port 80 et ssh sur le port 22.

Les éléments utiles nous sont donnés par **la commande ipconfig** : adresse IP, masque réseau, adresse MAC. nous verrons dans le cours sur le réseau que grâce au masque on pourra découper un réseau en plusieurs sous-réseaux par exemple découper un réseau de 255 machines en 4 groupes de 64 machines.

Le serveur SSH fonctionne en tant que service lancé automatiquement au démarrage de la machine sur notre raspberry pi. Sous Linux on peut ouvrir jusqu'à 6 terminaux (tty1, ..., tty6), pour ouvrir sur un terminal il suffit de taper :

CRTL + ALT + F1 ou CRTL + ALT + F2 ou .... CRTL + ALT + F6

Aller sur un terminal permet d'être plus proche du noyau. Pour revenir sur un terminal graphique : ALT + F7

Dans un terminal les commandes suivantes suivantes ont été saisies sur la raspberry (serveur : IP = 192.168.43.123) pour que la connexion d'un client soit possible :

sudo systemctl start ssh

Exemple : je suis sur une machine (IP = 192.168.43.12) et je désire placer ma page d'accueil index.html sur le serveur web de ma raspberry pi.

Sachant que les fichiers html du serveur se trouvent dans le répertoire var/www/html, nous aurons dans le terminal de ma machine cliente :

admin302@pc_prof : $ **ssh pi@192.168.43.123**

pwd : (à renseigner)

pi@192.168.43.123 : $ **cp index.html /var/www/html**

L'installation d'un logiciel sur linux se fait avec **la commande sudo apt install mon_paquet**

8 Les Processus - Ordonnancement

Nous avons vus en TP, comment tuer un processus :
la commande ps -alx | ps-aux
(*voir aussi : la commande top*)
nous permet de voir tous les processus dans les champs :
F, UID, PID, PPID, PRI, NI, VSZ, RSS, WCHAN,STAT,TTY, TIME, COMMAND.
Pour informations, voici un descriptif rapide des champs :

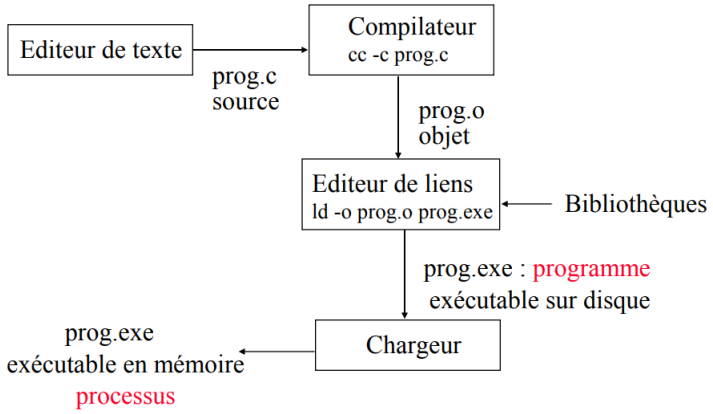
- F : champ d'indicateurs sur l'état du processus (création, arrêté par un signal),
- UID : UserIDentifier, le propriétaire du processus,
- PID : Process IDentifier, donne le numéro du processus,
- PPID : Parent Process IDentifier, PRI : fréquence des activations possibles,
- NI : Nice, indique la priorité, une valeur positive correspond à un moindre accès au processeuravec des valeurs entre -20 et 19,
- VSZ : taille virtuelle de l'image du processus(code+données+pile),
- RSS : taille en ko du processus en mémoire,
- WCHAN : Wait CHANnel donne le nom de la fonction noyau dans lequel le processus est endormi,
- STAT : donne l'état du processus Running/Sleeping/Dormant/Traced ou stopped/ sWapped/Zombie,
- TTY : identifie le terminal as1socié,
- TIME : donne le temps CPU consommé depuis le lancement, COMMAND : affiche le nom de la commande qui a lancé le processus (lorsqu'on a k : processus lancés par le noyau et les daemons sont suffixés par d : Disk And Exécution MONitor sont des processus lancés par le système attaché à aucun terminal : voir cron.d pour la Raspberry Pi) .

Si libreOffice writer est ouvert avec le PID 65211 que je désire l'arrêter car il a "planté" alors il faudra taper **la commande kill 65211** | **la commande kill -9 65211** si l'application ne répond plus .
Un processus est :

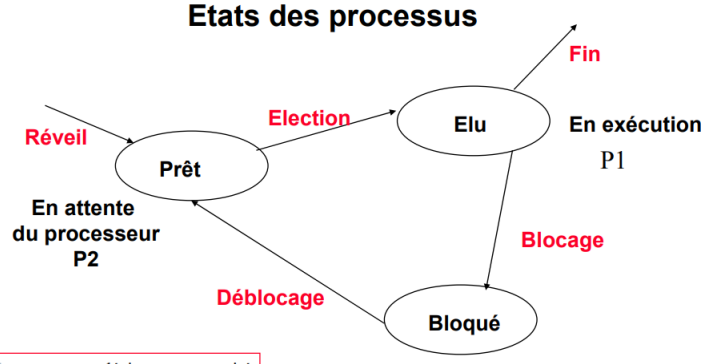
- Un processus est un programme en cours d'exécution auquel est associé un environnement processeur (CO, PSW, registres généraux) et un environnement mémoire appelés contexte du processus.
- Un processus est l'instance dynamique d 'un programme et incarne le fil d 'exécution de celui-ci
- Un processus évolue dans un espace d'adressage protégé

Les fonctionnalités du système d'exploitation sont accessibles par le biais des commandes ou des appels système.

Du programme au processus



Système multiprocessus
Etats des processus



Une ressource désigne toute entité (matérielle, logicielle) dont a besoin un processus pour s'exécuter (processeur, variable)

bloc de contrôle de processus PCB

identificateur processus
état du processus
compteur instructions
contexte pour reprise (registres et pointeurs, piles,..)
pointeurs sur file d' attente et priorité(ordonnancement)
informations mémoire (limites et tables pages/segments
informations de comptabilisation et sur les E/S, périphériques alloués, fichiers ouverts,..

Bloc de contrôle de processus ou PCB

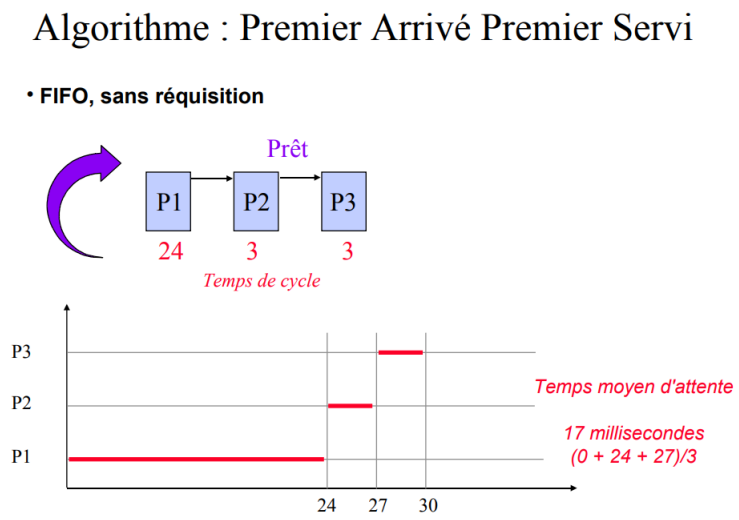
9 Ordonnancement

La gestion de l'accés au processeur est appelé ordonnancement (sheduling). A l'instant t, seul un processus peut disposer du processeur, mais alors lequel ? Un processus élu va-t-il garder le processeur jusqu'à la fin du programme ? Va-t-il le rendre ou faut-il que le système le récupère ?

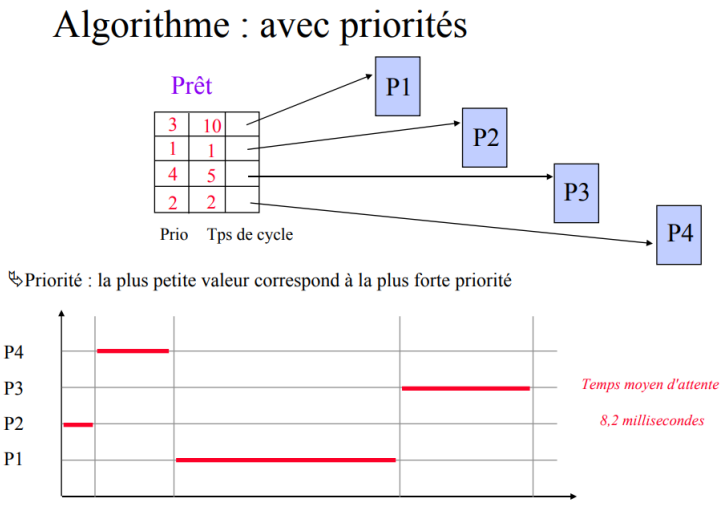
C'est l'ordonnanceur (scheduler) du noyau qui va répondre à ces questions selon la politique d'ordonnacement choisie. Sous Linux, cet ordonnanceur est implanté par la fonction schedule , voir /usr/src/linux/kernel/sched.c

Nous allons voir 3 algorithmes :

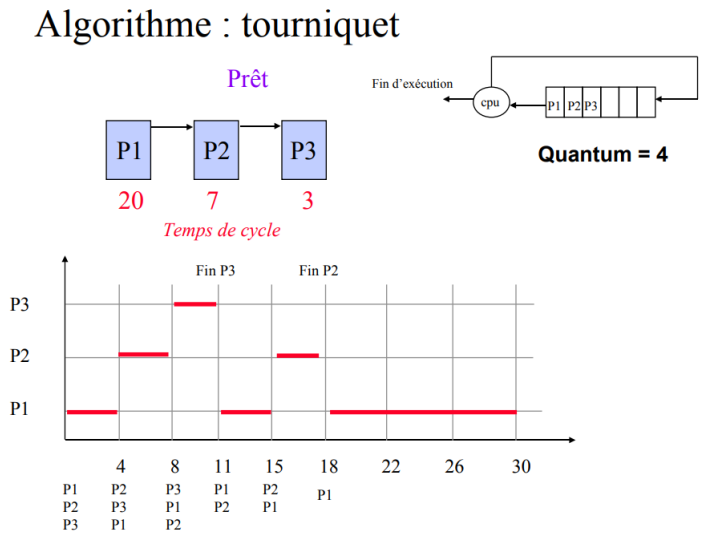
1. - Premier arrivé, premier servi FIFO, sans réquisition



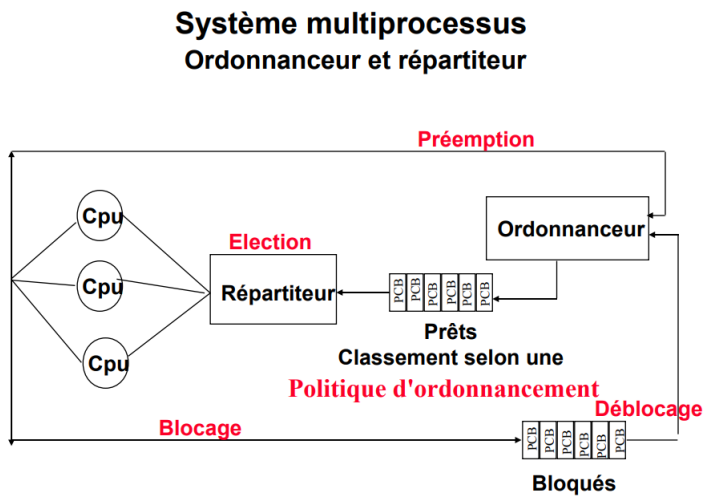
2. - Par priorités constantes : chaque processus reçoit une priorité, le processus de plus forte priorité est élu, avec ou sans réquisition :



3. - Par tourniquet (Round Robin) : Un quantum = tranche de temps, un processus élu s'exécute au plus durant un quantum ; à la fin du quantum, préemption et réinsertion en fin de file d'attente des processus prêts :

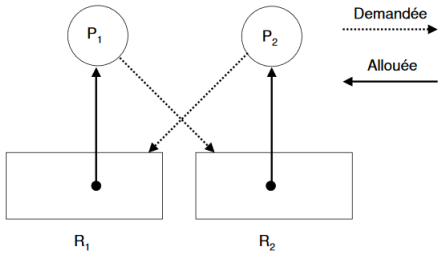


S'il y a plusieurs processeurs, la politique est la même mais un répartiteur gèrera l'envoie des quantum de processus dans chacun des processeurs :



10 Interblocage :

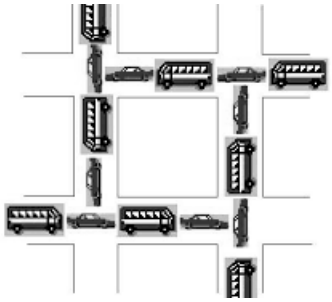
Des problèmes peuvent survenir lorsque les processus obtiennent des accès exclusifs aux ressources. Par exemple, un processus P_1 détient une ressource R_1 et attend une autre ressource R_2 qui est utilisée par un autre processus P_2 ; le processus P_2 détient la ressource R_2 et attend la ressource R_1. On a une situation d'interblocage (**deadlock** en anglais) car P_1 attend P_2 et P_2 attend P_1. Les deux processus vont attendre indéfiniment comme montré sur la figure ci-dessous. En général, un ensemble de processus est en interblocage si chaque processus attend la libération d'une ressource qui est allouée à un autre processus de l'ensemble. Comme tous les processus sont en attente, aucun ne pourra s'exécuter et donc libérer les ressources demandées par les autres. **Ils attendront tous indéfiniment !**



La résolution des interblocages constitue un point théorique très étudié.

La plupart des Os ont choisi de ne pas éviter les interblocages mais de les détecter s'ils surviennent et de les solutionner par un système de verrous, par exemple.

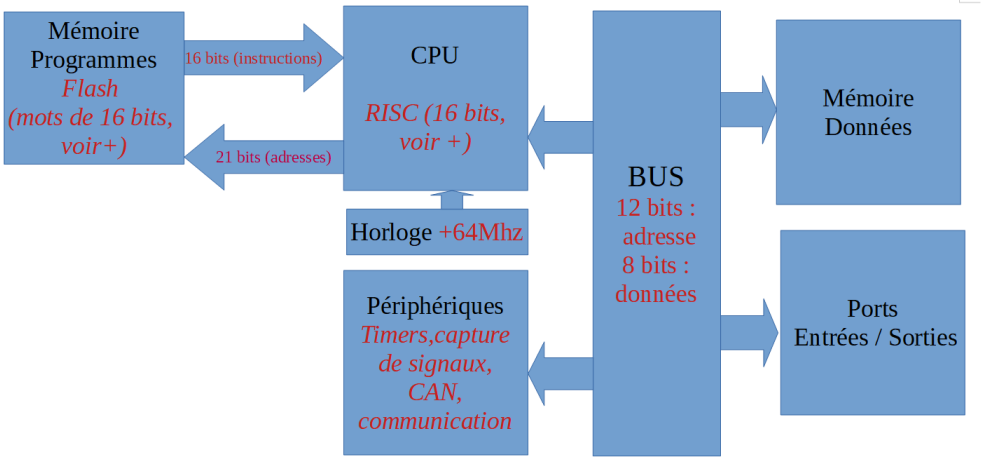
Voici un exemple un interblocage sur un réseau routier :



L'interblocage est le grand danger de la programmation concurrente (ou transactions concurrentes). Il y a 4 conditions nécessaires à la présence d'interblocage énoncées en 1971 et nommées **les conditions de Coffman** :

1. - *Exclusion mutuelle* : au moins une ressource du système doit être en accés exclusif.
2. - *Rétention et attente* : un processus detient une ressource et demande une autre ressource pour poursuivre détenue par un autre processus.
3. - *Non préemption* : une resssource ne peut être rendue que par un processus qui la détient (ne peut être acquise de force par un autre processus).
4. - *Attente circulaire* : un ensemble de processus bloqués attendent une ressource tenue par un autre ensemble de processus qui détiennent la ressource des processus bloqués car eux attendent une ressource des processus demandeurs.

11 L'architecture d'Havard pour les microcontôleurs et les Systems on Chip (SoC) :

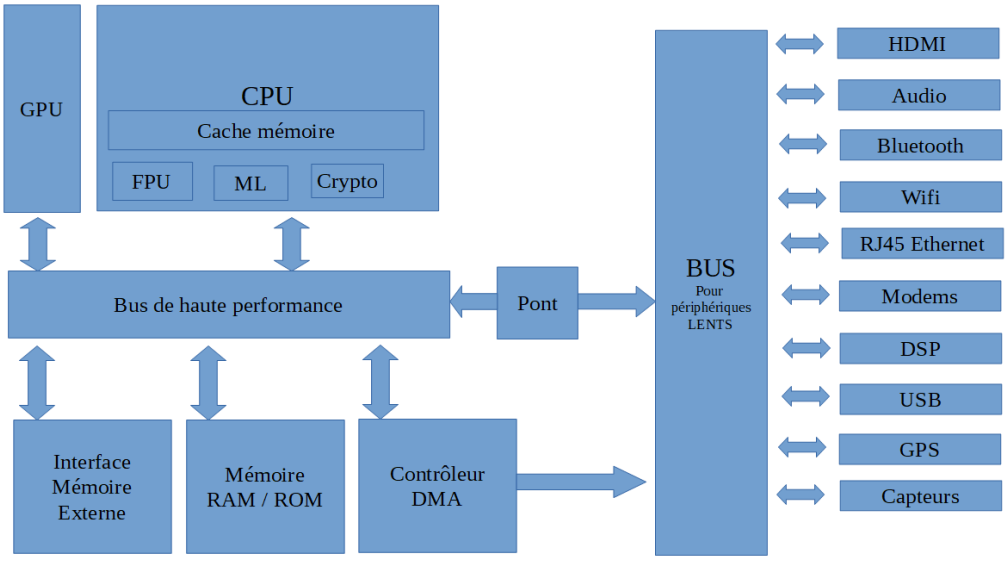


Architecture de HAVARD : exemple de la famille PIC (18F,24,32,...)

Ces circuits intégrés sont surtout utilisés dans des systèmes d'informatique embarqués (voitures, avions, machines à laver, robots,...). Ces circuits sont principalement des composants autonomes qui dès qu'ils sont mis sous tension exécutent le programme qui est contenu dans leur mémoire. La grosse différence avec l'architecture de Von Neumann vu en 1ère NSI est que la mémoire est séparée pour le programme et les données.

La mémoire Programmes est une mémoire morte : le programme est conservé même après une coupure de courant (type flash ou ROM ou EEPROM). La mémoire Données peut être de type morte ou RAM.

Le CPU a un jeu d'instructions réduit RISC (Reduced Instructions Set Computer) composé d'instructions simples, de longueurs fixes.



Architecture d'un System on Chip : SoC

Le SoC rassemble sur un même circuit intégré tous les composants que l'on trouve habituellement sur la carte mère d'un ordinateur. Le CPU est de type RISC mais en général il contient en plus des dédiés pour accélérer des calculs ou des opérations.

On peut trouver :

— un FPU : unité de calcul pour nombres flottants simple ou double précision.

— un circuit pour des opérations sur les matrices pour accélérer les algorithmes d'IA (ML : Machine Learning)

— une unité pour accélérer les algorithmes de cryptologie

Pour information :

les modems sont actuellement en 5G, des capteurs CDD (photo) avec un controleur DMA (Direct Memory Access) qui soulage le CPU.

Exemples :

le système sur puce *A13 bionic*, disponible sur les téléphones Apple est un SoC. Sa puce électronique contient 8.5 milliards de transistors gravés avec une précision de 7 nanomètres.

le système sur puce *A15 bionic*, disponible sur les téléphones Apple est aussi un SoC. Sorti en septembre 2021, sa puce électronique, ARM 64 bits de 15 milliards de transistors avec un moteur neuronal à 16 coeurs !