المدرسة العليا للتكنولوجيا - كلميم
ⵜⴰⵙⴷⴰⵡⵜ ⵜⴰⵏⵣⵣⴰⵢⵜ ⵏ ⵜⵙⵙⵏⴰⵡⵉⵏ - ⴽⵍⵉⵎⵉⵎ
Ecole Supérieure de Technologie - Guelmim

Cours
Administration des Systèmes d'Exploitation
Ecole Supérieure de Technologie – Guelmim
Université Ibn Zohr –Agadir-

DUT-Informatique –S3

Prof. ASIMI Younes

asimi.younes@gmail.com

2020/2021

Utilisateurs, groupes et autorisations

Classes d'utilisateurs et de Groupes

- ***User (l'utilisateur)***
  - le propriétaire du fichier
  - il est désigné par la lettre ***u***

- ***Group (groupe)***
  - les membres du groupe associé au fichier. il est
  - désigné par la lettre ***g***

- ***Others (autres)***
  - Ce terme générique désigne tous les utilisateurs autres que les membres du groupe et les propriétaires
  - ils sont désignés par la lettre ***o***

Utilisateurs, groupes et autorisations

Autorisations

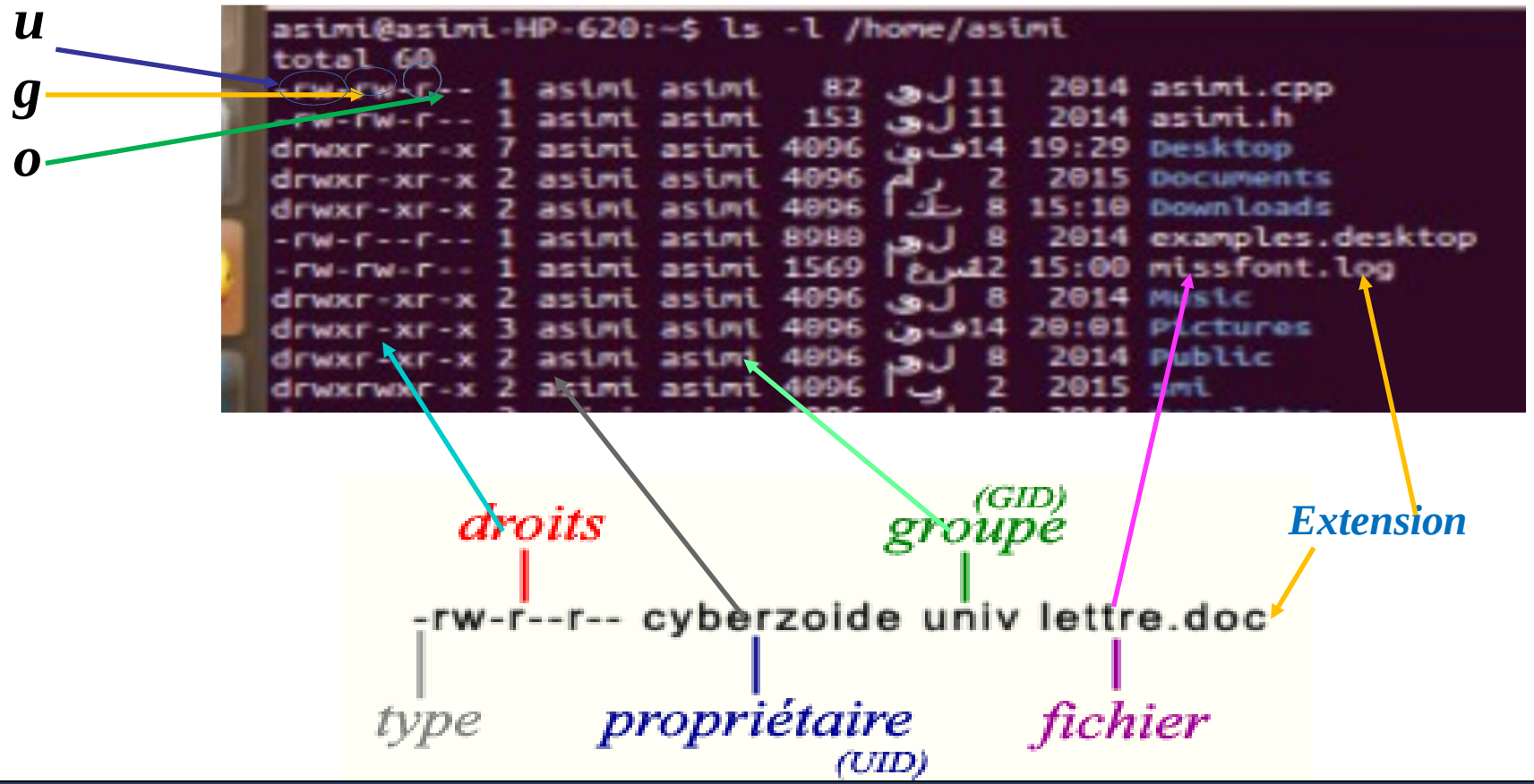
- Pour chaque classe d'utilisateurs, il y a 3 types d'accès à un fichier donné :
  - ***Read*** (*r*) : en **lecture**
  - ***Write*** (*w*): en **écriture**
  - ***eXecute*** (*x*) : en **exécution**

- Au niveau répertoire, ces droits signifient :
  - ***Read*** : droit de **lister** les fichiers présents dans ce répertoire
  - ***Write*** : droit de **créer** ou de **détruire** un fichier qui s'y trouve
  - ***eXecute*** : droit de **traverser** ce répertoire

- ***Rq: 3 types d'accès et les 3 classes d'utilisateurs, il y a donc 9 droits d'accès différents***

Utilisateurs, groupes et autorisations

Classes d'utilisateurs



Protections, Privilèges, Autorisations

La protection d'un fichier ne peut être modifiée que par le propriétaire à l'aide de la commande ***chmod*:**

Syntaxe: > **chmod 567 toto**

Binaire	Logique	Décimal
000	()	0
001	(x)	1
010	(w)	2
011	(wx)	3
100	(r)	4
101	(rx)	5
110	(rw)	6
111	(rwx)	7

On a donc rwx rw rx = 111 110 101 = 765

Protections, Privilèges, Autorisations

Le deuxième mode d'utilisation de **chmod, le mode symbolique:**

Syntaxe: ***chmod [who]op[permission] fichier***

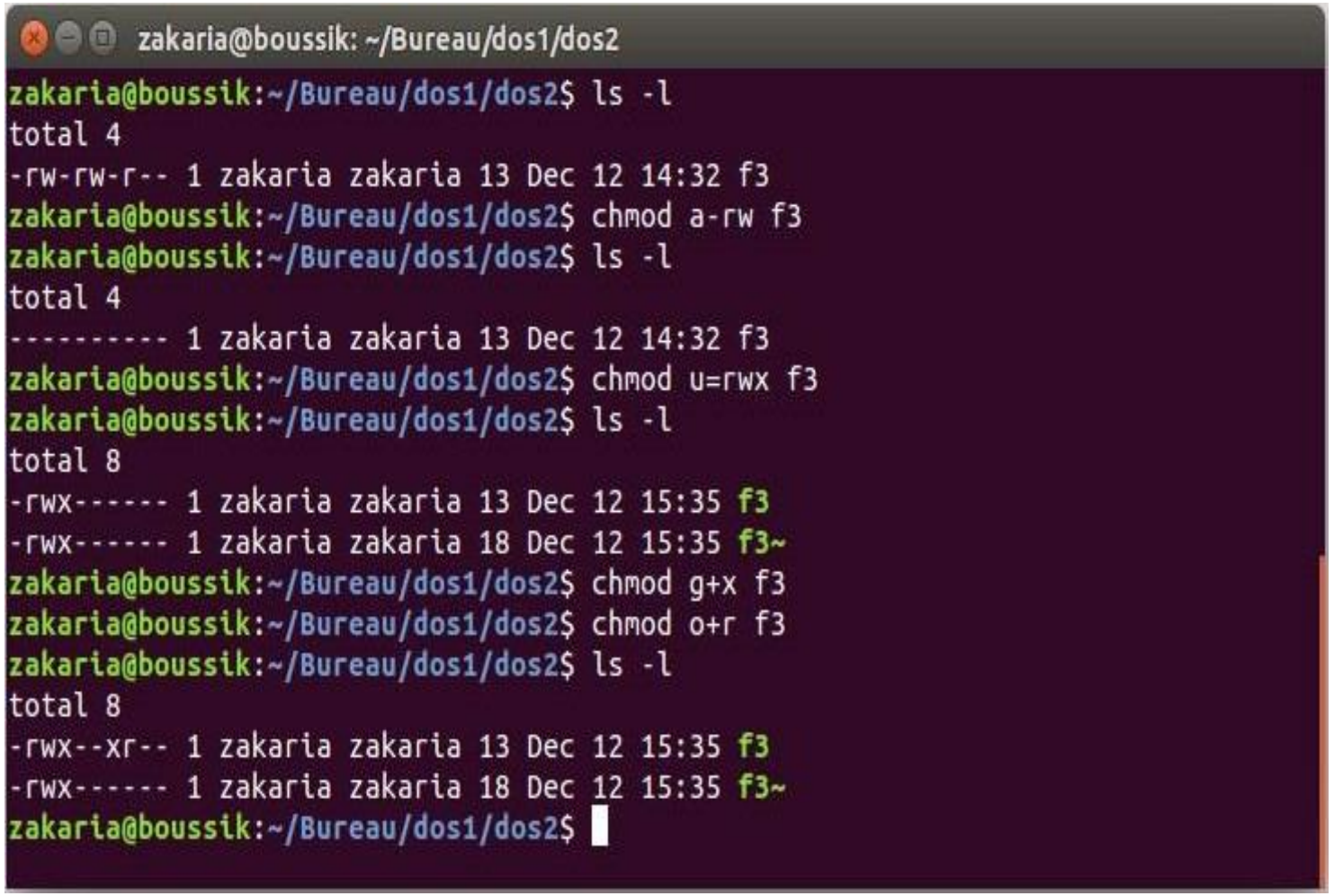
who	**lettre u (user=propriétaire), g (groupe), o (other=autre) ou a (all=tous) pour ugo,**
op	+ permet d'ajouter, - de supprimer et = d'affecter un droit de manière absolue.
permission	r w x

- **chmod u-w f1**
- **chmod g+r f2**
- **chmod ug=x f3**

Protections, Privilèges, Autorisations

```
zakaria@boussik: ~/Bureau/dos1
zakaria@boussik:~/Bureau/dos1$ ls -l
total 20
drwxrwxr-x 2 zakaria zakaria 4096 Dec 12 14:33 dos2
-rw-rw-r-- 2 zakaria zakaria   33 Dec 12 14:29 f1
-rw-rw-r-- 1 zakaria zakaria   13 Dec 12 14:30 f2
-rw-rw-r-- 2 zakaria zakaria   33 Dec 12 14:29 f5
-rw-rw-r-- 1 zakaria zakaria    0 Dec 12 15:03 f6
-rw-rw-r-- 1 zakaria zakaria   46 Dec 12 15:15 fs
zakaria@boussik:~/Bureau/dos1$ chmod 753 f1
zakaria@boussik:~/Bureau/dos1$ ls
dos2  f1  f2  f5  f6  fs
zakaria@boussik:~/Bureau/dos1$ ls -l
total 20
drwxrwxr-x 2 zakaria zakaria 4096 Dec 12 14:33 dos2
-rwxr-x-wx 2 zakaria zakaria   33 Dec 12 14:29 f1
-rw-rw-r-- 1 zakaria zakaria   13 Dec 12 14:30 f2
-rwxr-x-wx 2 zakaria zakaria   33 Dec 12 14:29 f5
-rw-rw-r-- 1 zakaria zakaria    0 Dec 12 15:03 f6
-rw-rw-r-- 1 zakaria zakaria   46 Dec 12 15:15 fs
zakaria@boussik:~/Bureau/dos1$ chmod ug=r-x f6
zakaria@boussik:~/Bureau/dos1$ ls -l
total 20
drwxrwxr-x 2 zakaria zakaria 4096 Dec 12 14:33 dos2
-rwxr-x-wx 2 zakaria zakaria   33 Dec 12 14:29 f1
-rw-rw-r-- 1 zakaria zakaria   13 Dec 12 14:30 f2
-rwxr-x-wx 2 zakaria zakaria   33 Dec 12 14:29 f5
-r--r----x 1 zakaria zakaria    0 Dec 12 15:03 f6
-rw-rw-r-- 1 zakaria zakaria   46 Dec 12 15:15 fs
zakaria@boussik:~/Bureau/dos1$
```

Protections, Privilèges, Autorisations



Protections, Privilèges, Autorisations

L'**umask** est un outil de gestion des droits d'accès aux fichiers et aux répertoires. Il permet de gérer la sécurité ou bien la restriction des permissions par défaut.

- Pour un fichier les permissions par défaut sont « **666** » ou « **rw-rw-rw-** »;
- Un répertoire (dossier) se crée par défaut avec les permissions « **777** » ou « **rwx rwx rwx**»;
- On peux appliquer la commande **umask** pour enlever automatiquement certains droits.
- Pour afficher le masque par défaut on exécute la commande suivante:
  - asimi@asimi-HP-620 ~ $ **umask -p**
  - umask 022
  - Rq: p = print

Protections, Privilèges, Autorisations

Le SE protège les fichiers et les dossiers par **un masque par défaut** « **022** » :

- asimi@asimi-HP-620 ~ $ ls -l

  total 56

  drwxr-xr-x 2 asimi asimi 4096 Oct 25 18:20 asimiDRWX

  -rw-r--r-- 1 asimi asimi 7787 Oct 25 18:15 asimiRWX.odt

- asimi@asimi-HP-620 ~ $

L'**umask** qui est par défaut défini pour les fichiers comme pour les répertoires est en fait « **022** » ce qui donne :

- pour les fichiers: **666-022 = 644 ou bien « rw-r--r--».**
- pour les répertoires : **777-022 = 755 ou bien « rwxr-xr-x ».**

Protections, Privilèges, Autorisations

Il est possible de modifier l'**umask** de façon temporaire, c'est-à-dire que nous allons mettre en place un **umask** avec une commande :

- **Syntaxe: > umask masque**
- **umask u=rwx, g=rx, o=rx**
- **umask 222**

- La valeur 132 est soustraite de la permission permanente (111 111 111) :

  111 111 111 < = permission permanente;

  001 011 010 < = on enlève les bits dont on ne veut pas ;

  110 100 101 = > 645;

- **umask 132** permet de créer des répertoires dont la protection est: rw- r-- r-x.
- Trouver les permission pour un fichier créé avec ce masque temporaire;
- ***Rq* importante: Le masque se réinitialise à sa valeur par défaut (022) à la fermeture d'une session.**

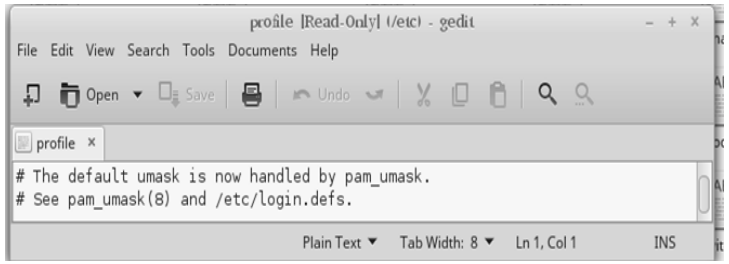
Protections, Privilèges, Autorisations

```
zakaria@boussik: ~/Bureau/a1
zakaria@boussik:~/Bureau$ umask 645
zakaria@boussik:~/Bureau$ touch f1
zakaria@boussik:~/Bureau$ ls -l
total 16
drwxrwxr-x 2 zakaria zakaria 4096 Dec 11 14:40 a1
drwxrwxr-x 2 zakaria zakaria 4096 Dec 11 15:06 coller
drwxrwxr-x 3 zakaria zakaria 4096 Dec 12 16:30 dos1
-----w--w- 1 zakaria zakaria    0 Dec 13 16:00 f1
drwxrwxr-x 4 zakaria zakaria 4096 Dec 13 15:54 Shell
zakaria@boussik:~/Bureau$ alias rp="echo 1-"
zakaria@boussik:~/Bureau$ rp bonjour
1- bonjour
zakaria@boussik:~/Bureau$ pwd; ls; rp salut
/home/zakaria/Bureau
a1  coller  dos1  f1  Shell
1- salut
zakaria@boussik:~/Bureau$ ls
a1  coller  dos1  f1  Shell
zakaria@boussik:~/Bureau$ cd a1 || ls
zakaria@boussik:~/Bureau/a1$ cd a1 && ls
bash: cd: a1: No such file or directory
zakaria@boussik:~/Bureau/a1$ cd ..; cd a1 && ls
f1  f2  f3  fi  if
zakaria@boussik:~/Bureau/a1$
```

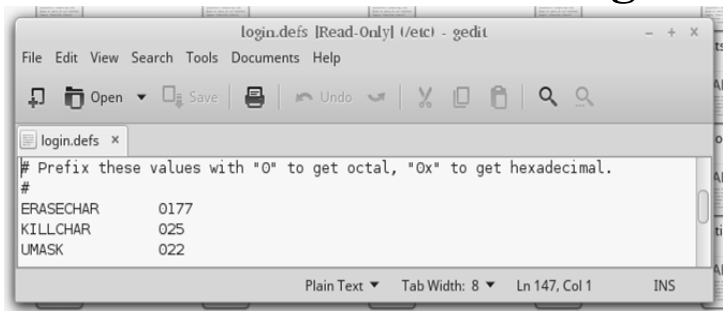
Protections, Privilèges, Autorisations

Changement du masque par défaut

Si on pense que le masque par défaut défini par le système ne réponds pas à notre exigence de point de vu sécurité. On peux redéfinir un nouveau masque. Par défaut, ce paramètre se gère dans « **/etc/profile** » :



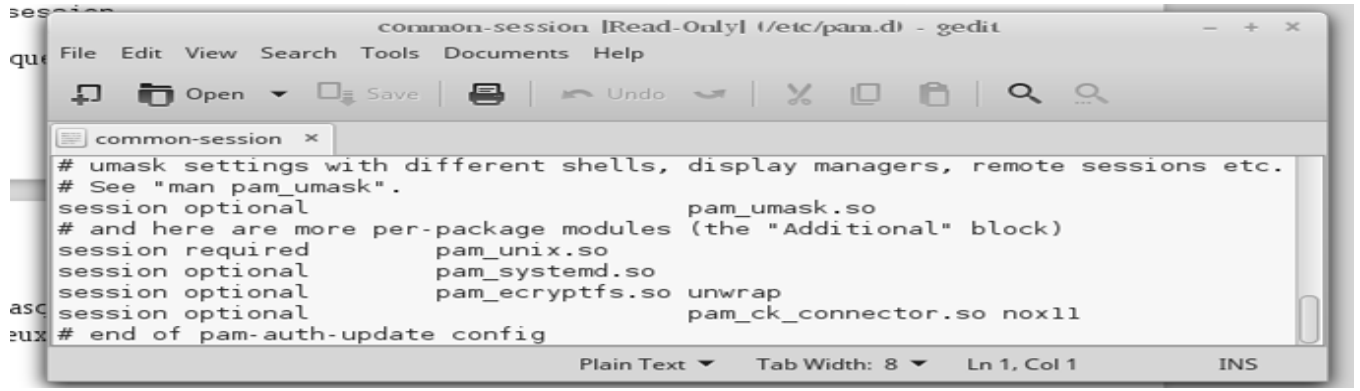
Nous voyons que le système nous dit que ce paramètre est maintenant géré par « **pam_umask** ». Nous devons voir le fichier « **/etc/login.defs** » pour modifier ce paramètre à la ligne **144**:



Protections, Privilèges, Autorisations

On doit ensuite dire au fichier de configuration PAM « **/etc/pam.d/common-session** » d'utiliser la module « **pam_umask.so** » qui gérera l'**umask** par défaut en lui ajoutant cette ligne :

```
# utilisation de la librairie umask
session optional pam_umask.so
```



Après redémarrage d'une session, nous pourrons vérifier que l'**umask** définie c'est bien appliqué avec la commande suivante : **umask -p**;

Processus

Présentation

Un processus est un programme en cours d'exécution. C'est-à-dire, un programme à l'état actif. Un processus regroupe:

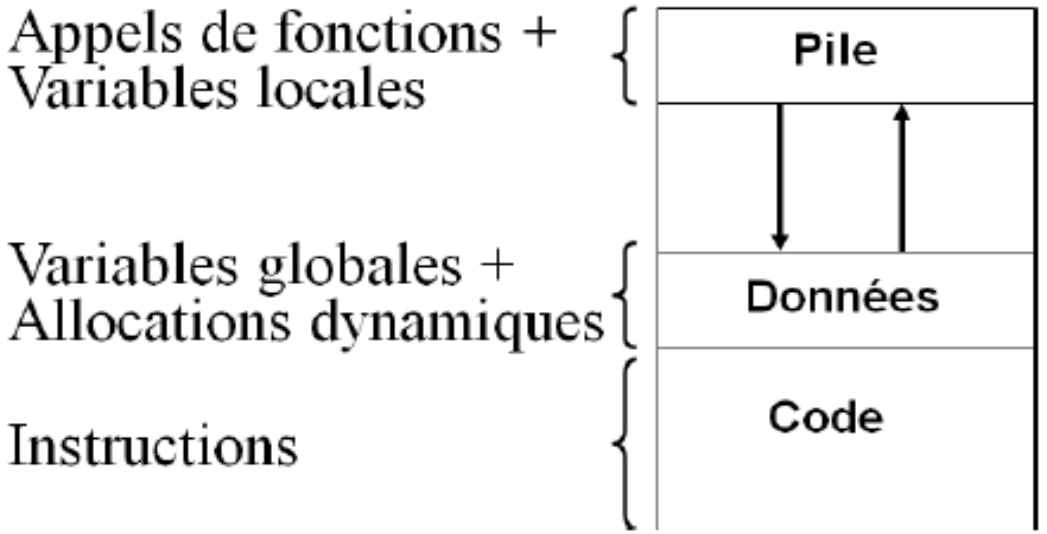
- Un programme exécutable;
- Sa structure de données privées (fichiers et Périphériques ouverts…);
- Son compteur ordinal (pour déterminer la prochaine instruction du programme);
- D'autres informations nécessaire à l'exécution du programme (**pile d'exécution, fichiers ouverts, etc.)**

- Un seul processus est exécuté à la fois sur un processeur (Si on a 2 processeurs, donc 2 processus en même temps…).
- Le processeur commute entre les différents processus (Ordonnancement);
- Le compteur ordinal permet de garder en mémoire la prochaine instruction à exécuter.

Processus

• **Un processus** est un programme en cours d'exécution qui est exécuté par un **microprocesseur**. Aussi, plusieurs processus peuvent-ils être associés à un programme. Chaque processus possède:

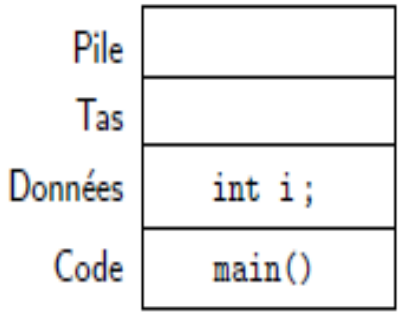
- Un espace de travail en mémoire;
- Un compteur ordina
- Des registres.



Espace de travail d'un processus

Processus

```
int i;
int main() {
  printf("Salut\n");
}
```



Source: Martin Quinson

- **Pile :** Garder trace de l'exécution de fonctions (fonctions appelantes, fonctions appelées)
- **Tas (Heap) :** Zone de mémoire dynamique (créer des variables durant l'exécution, allocation dynamique de mémoire)

Processus

Présentation

Un processus est une abstraction de l'exécution d'un programme par un processeur :

- ✓ Entité dynamique (Objet est une instance d'une classe);
- ✓ Evolue dans le temps;
- ✓ Passe d'un état initial à un état final;

Un programme est :

- ✓ Une entité statique;
- ✓ N'évolue pas dans le temps;
- ✓ Son contenu ne change pas malgré le changement du temps de la machine;

Processus

Point de vue utilisateur

- Lancer un programme exécutable:
  - Système crée un processus et gère son évolution;
- Possibilités de suivre un processus:
  - Connaître son état: **Progresse, Arrêté, Bloqué, Suspendu, …**
  - Agir sur le processus:
    - L'arrêter définitivement (Tuer le processus);
    - Le suspendre puis le reprendre;
- Exemples de processus:
  - Exécution d'une application;
  - Copier un fichier;
  - Envoi d'un message sur le réseau;
  - Lancer une impression.

Processus

Définition

Un processus n'est pas un programme exécutable.

Un processus correspond à une suite d'actions (notion dynamique) réalisées lors d'une exécution.

Processus

Création de processus

On a deux méthodes pour créer un processus:

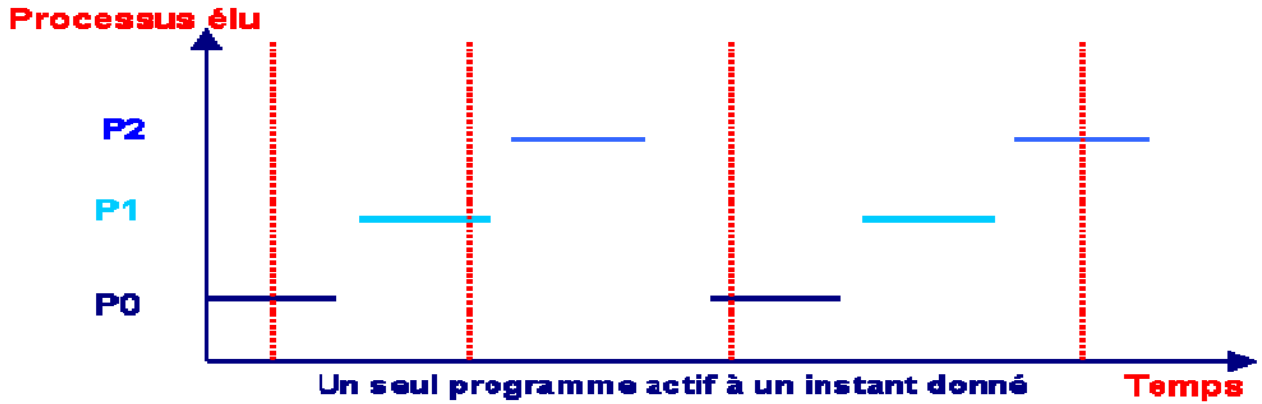
- Par lancement d'un programme:
  - **Commande ou click** sur l'icône du programme;
- Création dynamique par exécution d'instruction:
  - Un processus crée un autre et ainsi de suite;
  - **Fonction fork() et exec() (recouvrement) sous Unix (langage C);**
  - Méthode **start()** de la classe **Thread de JAVA(**unite de base qui partage les ressources avec processus) **;**

Processus

Commutation du processeur

- **Un** processus traduit l'image en mémoire centrale d'un programme s'exécutant avec son contexte d'exécution.

- Le processeur exécute successivement P0, P1, P2, P0, P1 et P2. Le processus en cours d'exécution se nomme: Processus élu.



- A un instant donné, le processeur n'exécute réellement qu'un seul programme. **La simultanéité n'est pas réelle;**

Processus

• Un Processus est un programme qui s'exécute et qui possède **son contexte d'exécution.** Durant sa vie, un processus peut:

- Se trouver dans différents états,
- Créer d'autre processus;
- Travailler et communiquer avec d'autres processus (Pipe);

• Différentes actions possibles d'un processus:

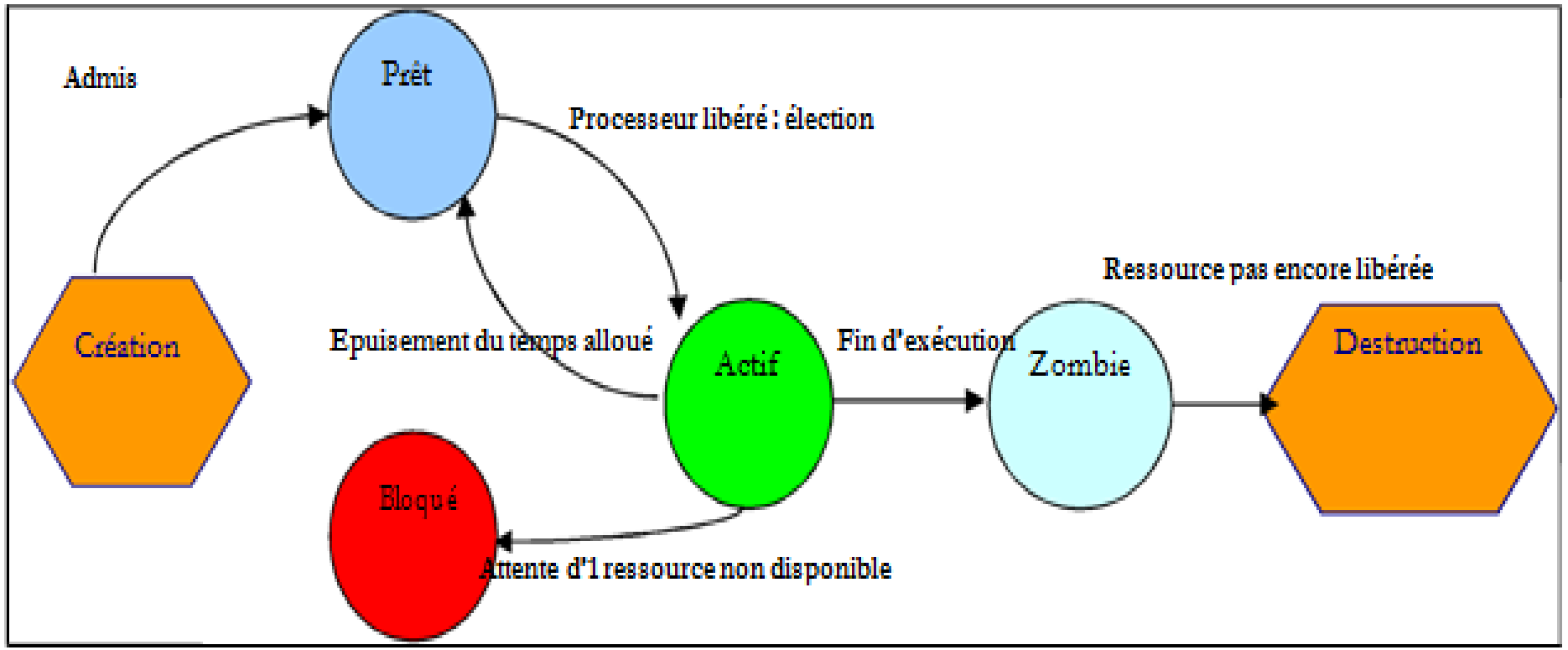
- **Un processus est créé.**
- **Un processus est Prêt :** le processus est en attente du processeur pour s'exécuter, ou bien, il suspendu provisoirement pour permettre l'exécution d'un autre programme (**priorité**).
- **Un processus actif :** le processus contrôle le microprocesseur central et s'exécute;

Processus

- **Un processus s'exécute:** il exécute les actions du programme dont il est l'image en mémoire centrale.
- **Un processus est bloqué** : Le processus est en attente d'une ressource pour terminer. Dès sa libération , il repasse à l'état Prêt.
- **Un processus est passif**: il n'a plus le contrôle du processeur central.
- **Un processus est détruit.**
- **Un processus est zombie :** Le processus a terminé son exécution et il ne peut plus évoluer, mais, les ressources qu'il a allouées ne sont pas encore libérées

Processus

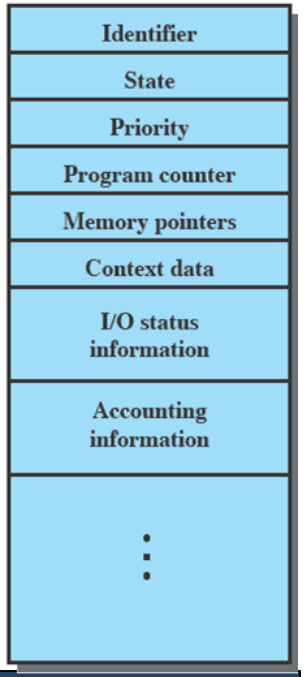
Cycle de vie d'un processus



Processus

• Bloc de Contrôle du Processus

• Pour implémenter les processus, le système d'exploitation utilise un tableau de structure, appelé **table des processus**. Cette dernière comprend une entrée par processus, allouée dynamiquement, correspondant au processus associé à ce programme : **Bloc de Contrôle du Processus (PCB)**.



• Il contient les informations suivantes :
- Le PID, le PPID, l'UID et le GID du processus ;
- L'état du processus (élu, bloqué…) ;
- Les données d'ordonnancement (Priorité, Temps);
- Les fichiers ouverts par le processus ;
- Le répertoire courant du processus ;
- Le terminal attaché au processus ;
- Les signaux reçus par le processus ;
- Information mémoire du processus.

Processus

• Bloc de Contrôle du Processus

- Les attributs d'un processus sont stockés dans une structure de données appelée **Bloc de Contrôle du Processus (PCB)**.
- Il contient suffisamment d'informations pour qu'il soit possible d'interrompre un processus en cours et de reprendre ultérieurement son exécution.

- Cette structure de données PCB du SE :
  - Contenir les informations sur un processus.
  - Ces informations sont dans une zone mémoire accessible uniquement par le noyau du SE.
  - Pour obtenir des informations sur les processus, il est donc nécessaire de passer par des **appels systèmes** :
    - **getpid, getppid, getuid, geteuid, setuid…**

Processus

• Attributs d'un processus

- Les attributs d'un **Bloc de Contrôle du Processus (PCB)** sont typiquement se scindés:
- Gestion de processus; Gestion de mémoire Gestion de fichiers

Registres	Pointeur sur code	Masque `umask`
Compteur ordinal	Pointeur sur données	Répertoire racine
État du programme	Pointeur sur pile	Répertoire de travail
Pointeur de pile	Statut de fin d'exécution	Descripteurs de fichiers
Date de création	N° de signal du proc. tué	UID effectif
Temps CPU utilisé	PID	GID effectif
Temps CPU des fils	Processus père	
Date de la proch. alarme	Groupe de processus	
Pointeurs sur messages	UID réel	
Bits signaux en attente	UID effectif	
PID	GID réel	
	GID effectif	
	Bits des signaux	

Processus

• Attributs d'un processus

- Un processus est caractérisé par:
  - État du processus:
    - Cela spécifie l'état du processus: nouveau, prêt, en cours d'exécution, en attente ou terminé.
  - Numéro de processus:
    - Cela montre le numéro du processus particulier.
  - Compteur de programme:
    - Il contient l'adresse de la prochaine instruction qui doit être exécutée dans le processus.
  - Registres:
    - Le processus utilise ces registres pour stocker les données: **des accumulateurs, des pointeurs de pile, …**

Processus

• Attributs d'un processus

- Liste des fichiers ouverts:
  - Ce sont les différents fichiers associés au processus;
- Informations de planification du processeur:
  - La priorité de processus, les pointeurs vers les files d'attente de planification, etc.
- Informations sur la gestion de la mémoire:
  - Les informations de gestion de la mémoire comprennent les tableaux de pages ou les tableaux de segments.
- Informations sur l'état des E / S:
  - Ces informations comprennent la liste des périphériques d'E / S utilisés par le processus, la liste des fichiers, etc.

Processus

• Attributs d'un processus

- Information comptable:
  - Les délais, les numéros de compte, la quantité de CPU utilisée, les numéros de processus, etc. font tous partie des informations comptables du PCB.
- Emplacement du bloc de contrôle de processus:
  - Le bloc de contrôle de processus est conservé dans une zone de mémoire protégée contre l'accès normal de l'utilisateur.
  - Il contient des informations de processus importantes.
  - Le SE place le PCB dans un emplacement sûr (la pile du noyau).

Processus

• Attributs d'un processus

• Nous pouvons regrouper les informations d'un **Bloc de Contrôle du Processus (PCB)** en trois catégories générales:

- Identification des processus:
  - Chaque processus est identifié par un numéro unique.
- Informations sur l'état du processeur:
  - Il s'agit du contenu des registres du processeur.
  - Registres visibles par l'utilisateur
  - Registres de contrôle et d'état
  - Pointeurs de pile …
- Informations de contrôle de processus:
  - Ces informations permettent au SE de **contrôler et coordonner** les différents processus actifs: **État du processus, priorité, identité de l'événement en attente, etc.**

Processus

- Un processus est identifié par un identifiant unique **PID (ProcessIDentifier) . Parmi ses i**nformations, on cite:
  - Le PID est attribué par le système à la création;
  - Le père d'un processus est identifié par son **PPID;**
- Un processus particulier (**Processus racine**) : le processus **init**
  - PID = 1
  - Initialise un terminal par utilisateur connecté à la machine;
  - Processus père de tous les interpréteur de commandes;

Processus

• Processus ou bien objet dynamique : qui représente un programme en cours d'exécution et son contexte:

- Caractéristiques:
  - identification (pid);
  - identification du proc. parent (ppid);
  - Propriétaire (UID utilisateur qui a lancé ce processus);
  - Priorité;
  - Privilège du propriétaire du fichier, Etat, ….
- Pour voir les processus en cours: **ps**

Processus

- Pour la gestion des processus sous Unix, cinq principaux appels système sont prévus. Il s'agit des appels dédiés à :

- **L'identification :**
  - **int getpid (void)** qui retourne l'identité du processus courant;
  - **int getppid(void)** qui retourne l'identité du processus père.
- **La création : int fork (void)** crée un processus fils et retourne:
  - -1 en cas d'erreur;
  - 0 dans le processus fils;
  - l'identité du fils dans le processus père.
- **La terminaison : void exit (int)** termine le processus appelant et met la valeur int dans son PCB.
- **L'attente : int wait (int *)** provoque une attente de la fin d'un processus fils; retourne -1 si le processus n'a pas de fils.

Processus

• La fonction fork()

• Cette fonction permet la création d' un nouveau processus à partir d'un programme. On distique deux niveau des processus:

- le processus d'origine est nommé processus père, qui possède un **PPID**;
- le nouveau processus créé processus fils, qui possède un nouveau **PID**.

• **La fonction fork()** retourne une valeur de **type pid_t.** Il s'agit généralement d'un **int** déclaré dans la bibliothèque **<sys/types.h>.** Il renvoyée la valeur :

- **-1** si il y a eu une erreur ;
- **0** si on est dans le processus fils ;
- **Le PID** du fils si on est dans le processus père.

Processus

• La fonction fork()

• Cette fonction crée un nouveau processus (processus fils) qui est une copie exacte du processus appelant (processus père).

- Le processus fils hérite du processus père :
  - La priorité;
  - La valeur du masque;
  - Le propriétaire;
  - Les descripteurs des fichiers ouverts (clé pour accéder à un fichier: stdin, stdout, stderr);
  - Le pointeur de fichier pour chaque fichier ouvert;
  - Le comportement vis à vis des signaux (Communication inter-processus).

Processus

• Autres fonctions

- La fonction **getpid** retourne le **PID** du processus appelant:
- **pid_t getpid(void);**
- La fonction **getppid** retourne le **PPID** du processus appelant:
- **pid_t getppid(void);**
- La fonction **getuid** retourne **l'UID** du processus appelant:
- **uid_t getuid(void);**
- La fonction **getgid** retourne le **GID** du processus appelant:
- **gid_t getgid(void);**

Création d'un Processus

*main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main () {
        printf (" Création d'un processus...\n");
        pid_t pid ;
        pid=fork();
        if ( pid == -1)
                fprintf (stderr , " fork   failed \n");
        else {
            if ( pid !=0) {
            /* c'est le fils */
                  printf(" Le PID du fils est %d.\n Le PPID du fils est %d.\n",  getpid(), getppid());
                   //exit (5);
                  // printf(" Le PID du fils est %d.\n Le PPID du fils est %d.\n", getpid(), getppid());
            }
        }
        }
```

Processus

File Edit View Search Terminal Help

```
Création d'un processus...
Le PID du fils est 4181.
Le PPID du fils est 4180.
Process returned 53 (0x35)   execution time : 0.002 s
Press ENTER to continue.
```