Processus

Un processus est un programme en cours d’exécution

Il dispose de son espace d’adressage propre

-Code de programme

-Données

-Pile

-Ensemble de registres

Table de processus

Plusieurs processus s’exécutant (presque en même temps) sur la une machine seront confrontés aux problèmes de concurrences

Gestion de la mémoire

->problème pagination

-Pas de solution idéal

- Solution acceptable

-FIFO

-LRU

-FWF

Système de fichier

->Notion racine

->Arborescence

->Chemin relatif/absolu

Appel système

Exemple : cpt= read(df,&temp,nboct) ;

//Insérer photo téléphone

Quelque appel système

->Processus: fork() waitpid() execute()

->fichier: open(),close(),write()

->Système de fichier :link()

->Signaux : kill()

Une exécution concurrente réel (2 ou plus processus en même temps) s’effectue sur une machine qui dispose de plusieurs processus

La création de processus a lieu lors :

-de l’utilisation du système

-de l’exécution de l’appel système de création de processus

-> requête utilisateur

-> De l’utilisation d’un traitement par lot

Les processus ont des identifiants

->pid : identifiant de processus (commande ps)

->uid : identifiant de l’utilisateur du processus

->gid : identifiant du groupe de l’utilisateur

Terminaison d’un processus

->Normale (exit(SUCCESS)/return 0 ;)

->Erreur (exit(FAILURE) ;)

->Erreur fatale( erreur de segmentation , Division par 0)

->Terminaison par un autre processus (la commande kill)

Gestion des processus

->premier nt

->un processus est crée par un autre processus

->arborescence de processus

->visualisation : ps , top , pstree

->suppression : kill

Voir création de processus

fork crée un processus copié du processus appelant.

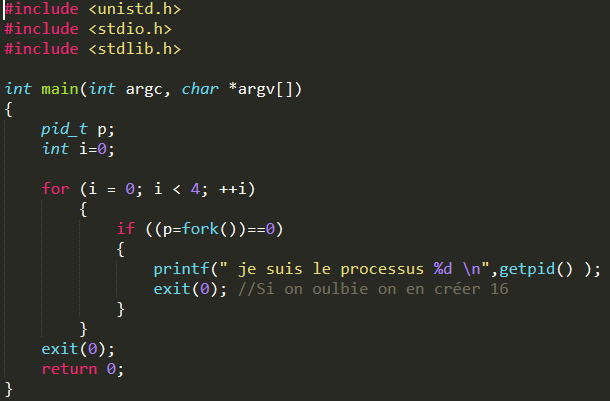
La valeur de retour est :

-0 dans le processus créé

- >0(id du processus créé) dans le processus appelants

Exercice 1:

Ecrire un programme qui va créer 4 processus fils, chaque processus devra afficher son ordre



Etat des processeurs

Les processus peuvent avoir les étais suivant :

-Exécution

-Sommeil(interruptible/non-interruptible)

-Arrêt

-Zombie

Exécution d’un programme

On utilise la famille d’appels système « exec »

int execv( const \*app , const char\* argv[]) ;

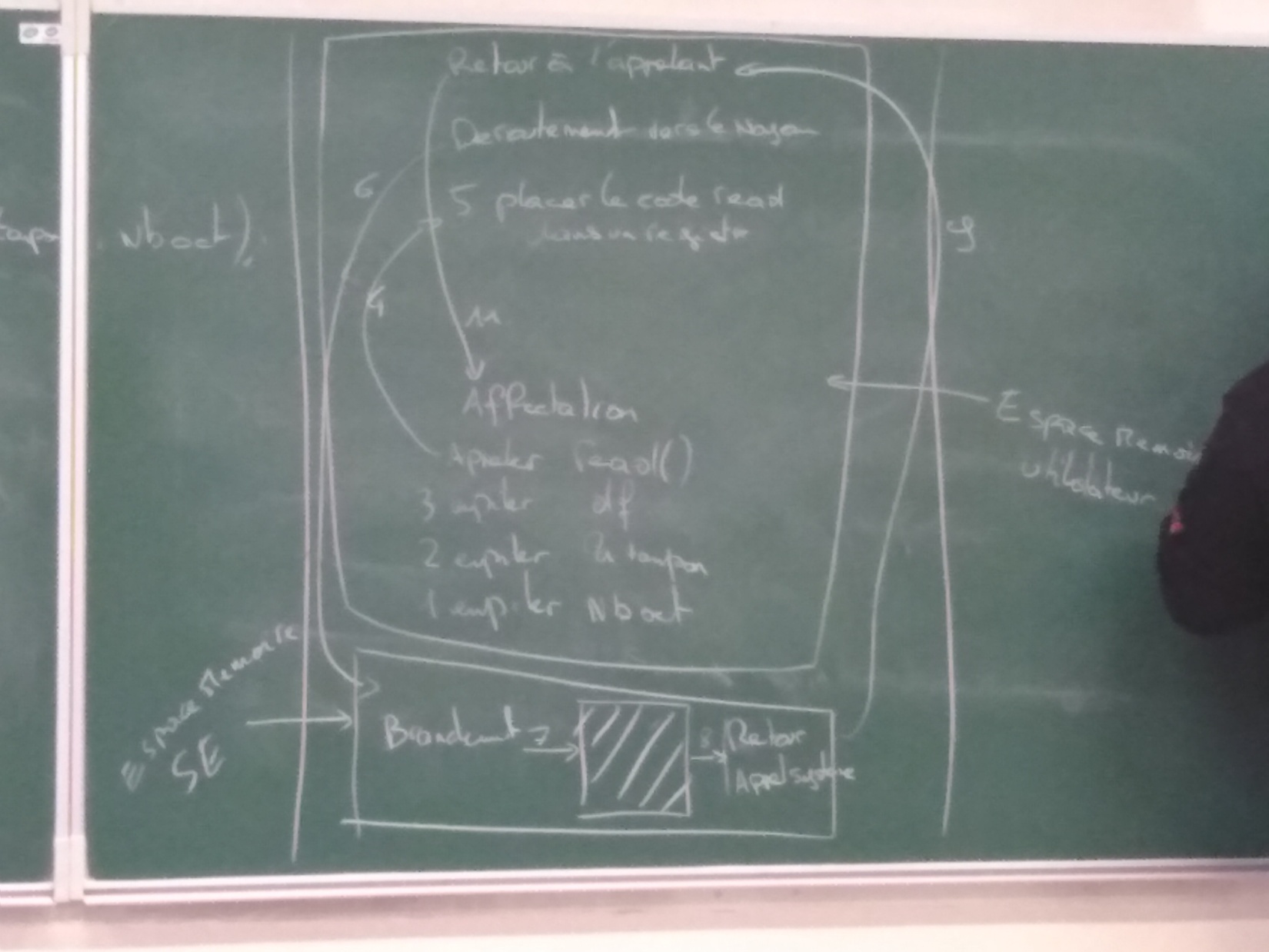
app : chemin complet du programme a exécuté

argv : paramètre à donner a l’application

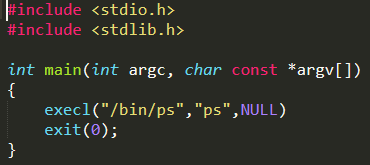
int execl( const \*app , const char\* arg1, cont char\* arg2,…) ;

app : chemin complet du programme a exécuté

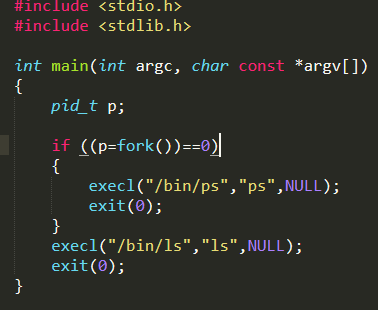
arg(n) :argument n



Exercice 2 : Ecrire un programme qui exécute ps



Exercice3 : Ecrire un programme qui exécute ps puis ls



Ordonnancement de processus

Motivation :

-Equité

-Gain de temps

Objectif équilibre

Critère pour un bon ordonnancement :

-Pourcentage d’utilisation CPU

-NB de processus exécuté en un temps donné

-Temps pour exécuter un processus

-Temps d’attente d’un processus dans la file prêt

-Temps de réponse

-…

Définition

L’ordonnanceur est un algorithme qui lit un processus. Le processus a le privilège d’accéder au CPU.

2 Types d’ordonnanceur :

-Non préemptif : Il sélection un processus qui s’exécute jusqu’à qu’il bloque ou qu’il libère volontairement le processus. (Terminaison).

-Préventif : Sélection un processus et l’exécute suivant un laps de temps donné. Si le processus est encore actif après ce délai, il est suspendu et l’ordonnanceur élit à nouveau un processus

Différent ordonnanceur :

Non préemptif

-FIFO: First In First Out

-SJF: Short Job First

-LJF: Longest Job First

Préemptif

-SRTN: Shortest Remaining Time Next (On choisit le processus avec le moins de temps restant)

-Round Robin (Notion de Quantum)

-Priorité : Par la file de priorité la plus haute

On considère 5 processus J1…J5

|  |  |  |  |
| --- | --- | --- | --- |
| job | Durée(sec) | Soumission(sec) | priorité |
| J1 | 5 | 1 | 4 |
| J2 | 6 | 3 | 5 |
| J3 | 10 | 0 | 1 |
| J4 | 7 | 3 | 4 |
| J5 | 6 | 2 | 5 |

FIFO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J3 | J1 | J5 | J2 | J4 |

10 15 21 27 34

SJF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J3 | J1 | J2 | J5 | J4 |

10 15 21 27 34

LJF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J3 | J4 | J5 | J2 | J1 |

10 17 23 29 34

SRTN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| J3 | J1 | J1 | J1 | J2 | J5 | J4 | J3 |

1 2 3 6 12 18 25 34