

# Chapitre 0 : Rappel sur les systèmes d'exploitation

**SVC**  
Superviser Call  
appels système

SOS  
SJS

Programmes

Schéduleur :  
programme qui  
gère la file d'att  
du processus

LIFO

TCS

Interruption

Prog  
Système

Programme User

Processus

Logiciels

TC2  
TC4

Materielles

TC3  
TC5

Routine d'IT  
(fork, write,  
fin prog...)

les apl sys  
(SVC)

PCB de processus

Contexte de processus

Etats de processus

→ TC1 : initialisation du schéma du diagramme

Prêt

Actif

Bloqué

→ TC2 : SVC (Dem EIS)

attend la  
libération  
du proc.

en cours

en attente  
d'exécution de ressource

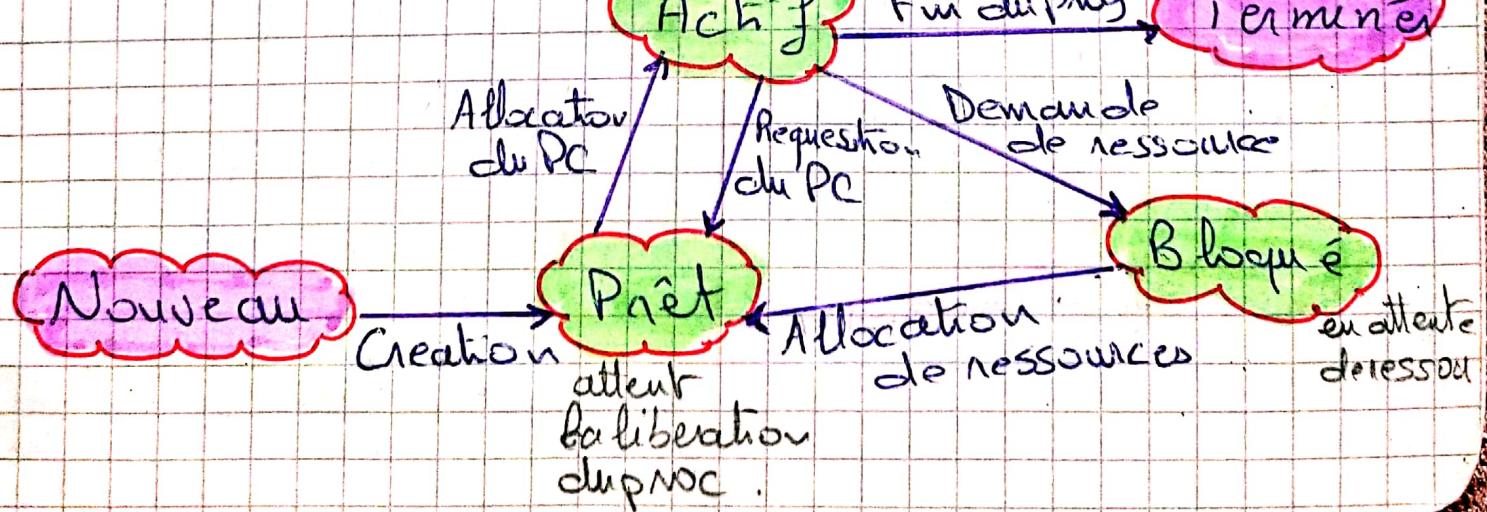
→ TC3 : Routine Fin EIS

→ TC4 : SVC (Fin processus)

→ TC5 : Routine fin quantum (Temps partagé)

Transition des états d'un processus :

en cours  
d'exécution



## Traitement des TCs:

SUC (cause,

<SC>

Switch cause of :

Dem EIS :

P-actif. etat = "Bloquee"; change l'état

Enfiler(P-actif, FA-B); mettre dans l'att d= bloquer

Si (Preiph libre) alors : lancer EIS

Si non :

mettre la dem en attente;

F.Si

LPSW (Scheduler); → activer le prochain processus.

Fin Processus :

Libérer les ressources allouées à P-actif;

LPSW (Scheduler); → activer le prochain processus.

<RCs

TC3:

Routine de Fin EIS

<SC>

P-actif. etat = "Pret";

Enfiler(P-actif, FA);

Si (FA vide) Alors

Débloquer des Dem enAtt;

F.Si

Reprendre le proc suspendu;

<RCs

TC5:

Routine de Fin Q

<SC>

P-actif. etat = "Pret";

Debloquer

Enfiler(P-actif, FA);

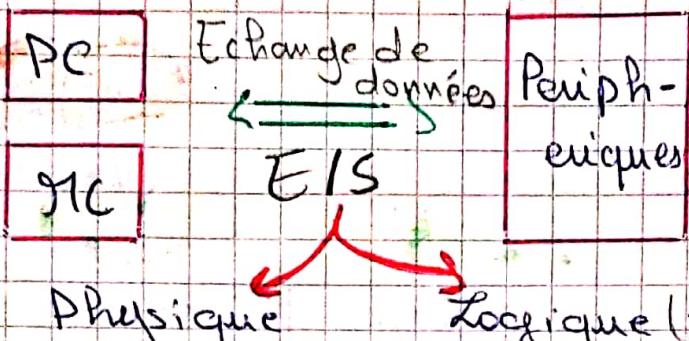
LPSW (Scheduler);

→ l'arrête q.

<RCs

## Chapitre 1:

### Gestion des E/S physiques.



E/S : échange de données

Dans la machine, l'E/S se transforme en E/S physique.

Logique (sous forme d'inst. print...) de haut niveau (Java, Python...)

## Le matériel :

### → Peripheriques :

#### Blocs

Accepte les données par blocs de taille fixe, avec chaque une adresse et un accès séquentiel.

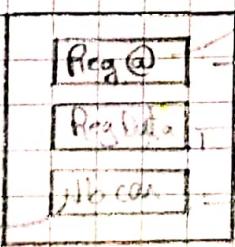
#### Caractères

Accepte les données en octets clairs, sans empêcher les terminaux.

### → Canaux : (I/O channels)

- gère un ou plusieurs périphériques
- Allège le travail du processeur principal pour la communication entre les contrôleurs

Canal DMA



à l'@ d'initialisation  
du donnée

initiale par ligne locutrice

Un périphérique a un seul canal

### → Contrôleurs

Registre Data transmis

Registre State occupé

Registre Control qui va

- ajuster la vitesse de communication
- lire les données du périphérique / PC / NIC
- contrôler, tester, détecter les erreurs du périphérique

### → Bus :

- Transforme les données
- chaque unité a sa propre capacité (unité mémoire...)

(processeur, mémoire...).

Un canal peut avoir plusieurs périphériques de même type

# Mode de pilotage

il faut passer par le processeur

Periphérique

Directe

Piloter

Synchronisé 100%

Piloter

suivre

Asynchrone 50%

Indirecte

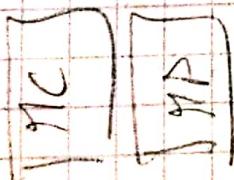
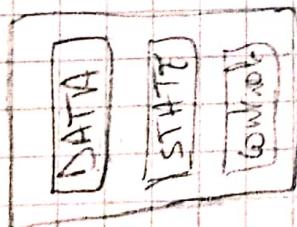
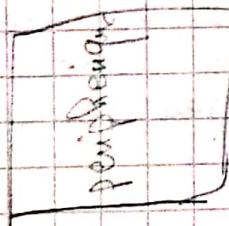
Piloter

suivre

DMA 1%

- Le processeur suit toute l'opération d'EIS en surveillant le registre d'état pour voir si l'op d'EIS est fini.

- Raige les info nécessaire puis lance le transfert après vérification si le periph est libre.
- Reste en attente jusqu'à la fin du transfert



- Le processeur lance l'EIS, puis à chaque fin de transfert il négocie une interruption pour l'avertir, entre temps le processeur exécute d'autre programme (scheduling).

Programmes:

SVC (cause, n, ...)  
(SCS)

Switch cause of:  
Dem EIS

P.actif.etat = "Blo";  
Enfilet(P.actif, FB);  
Si (periph pret):  
    Initialiser lancer1 en can;  
    Cpt = 1;  
Sinon:  
    mettre la demande en attente

(RCS) FSI

Routine Fin EIS d'un can  
(SCS)

Si (cpt > n) Alors:  
    lancer can suivant;  
    cpt = cpt + 1;  
Sinon: Si (b.can = 0)

    Enfilet(P.actif, FA);  
    Debloque l'op de la can en ATT  
    Reprendre le prog  
    Scheduler le progr  
    S'acheminer;  
(RCS) F.Si Debloque Dem en att

Direct Access

Mémoire

Le processeur initialise les registres du DMA puis ordonne le début de l'EIS et fait les autres tâches (hors d'EIS).

Nb = 0 : DMA envoie un signal de fin d'EIS  
PC: exécute la R.Fin EIS

Programmes:

SVC (cause, n, ...)  
(SCS)

Switch cause of:

Dem EIS (asynchronous DMA)  
P.actif.etat = "Blo";  
Enfilet(P.actif, FB);  
Si (canal pret):  
    Initialiser le canal,  
    SDO, Donner l'ordre  
    pour communiquer

Sinon:  
    mettre la demande en attente;

F.Si LPSW (scheduler);  
(RCS)

Routine fin EIS :

(SCS)

Enfilet (P.actif, FA);  
P.actif.etat = "pret";

Reprendre le prog scheduler;  
(RCS) L'op est pris par l'LPSW (x8)

(RCS) [PC] [canal DMA] [cont vol] [periph]

## Chapitre 2:

### Gestion du processus central.

#### Politique de scheduling :

→ Non Préemptives :

• FIFO (First In First Out) :

Appelé aussi FCFS (First Come First Serve)

- Premier arrivé, premier en service.

- Les programmes ne peuvent pas être interrompus par d'autres.

• SJF (Shortest Job First) :

- On commence par le processus le plus court.

- Quand un processus court arrive, on doit attendre que le processus en cours finisse son exécution, pour ensuite activer le processus le plus court.

- Le tri : se fait à l'intérieur du scheduler, ou par un programme spécial.

→ Préemptives :

• SRTF (Shortest Remaining Time First) :

- On fait la comparaison sur le temps d'exécution restant, à chaque arrivée d'un processus ou chaque unité

- Si les temps sont égaux, on laisse le processus en cours d'exécution.

ne peut pas être interrompu par d'autres proc (que par les I/Os)

un processus peut interrompre un autre

## • A base de priorité :

Selon l'ordre de priorité des processus données.

## • RR (Round Robin) :

C'est la politique scheduling du système d'exploitation à temps partagé.

## • MFP (MultiLevel Feedback Queues) :

Plusieurs files avec plusieurs politiques, différentes de pas.

Relier entre eux, de tel façon que chaque envoie vers l'autre.

TPs ont un ordre de priorité entre eux.

$$\text{Temps de réponse} = \text{Temps Fin d'exécution} - \text{Temps d'attente}$$

$$\text{Temps d'attente} = \text{Temps de réponse} - \text{durée d'exécution}$$

## Chapitre 3 :

### Gestion de la mémoire Centrale

#### Stratégie d'Allocation

Alloc à la fois

Application  
contiguë

un seul bloc  
à procéder  
à l'allocation

Alloc bloc à la fois

Défaut de  
succession  
des blocs

Partage  
multiples

taille de la portion

Partitions  
Fixes

taille  
fixe

Partitions  
variables

taille  
variable

Utilisation  
d'un seul  
fichier

Utilisation  
de plusieurs  
fichiers

chaque partition a sa  
fichier d'attente

Fragmentation interne:  
l'espace vide dans la MC  
lors de chaque partition  
qui affecte le niveau  
d'occupation

ou de divise

Application  
non contiguë

de deux lacs et le  
proc en page de  
la taille

Paginat ion

Segmentation

Segmentation  
Page

Fragm entation externe:  
l'espace libre dans la MC  
lors de l'allocation  
du proc qui lui convient

## Allocation contiguë

### Monobloc :

La MC contient deux partitions contigües :

- Une pour le système d'exploitation.

- Une pour l'utilisateur.

Opé alloue l'espace mémoire à un seul processus

atif à la fois.

### Partition multiples :

#### Partitions fixes :

La MC est partagé en plusieurs partition de taille fixe et pas forcément égal.

#### → Utilisation de files multiples :

Chaque partition a sa propre file d'attente.

Chaque processus à son arrivé est placé dans la file d'attente de la partition la plus petite qui le suffit.

#### → Utilisation d'un seul file :

Toute la MC a une seule file d'attente.

Chaque fois qu'une partition se libère, on lui affecte processus de la file qui peut y tenir.

### Partitions variables :

atouve exactement la taille du processus.

@ basses

MC

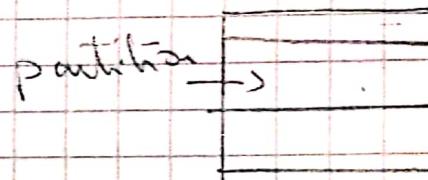
Système  
d'exploitation

User

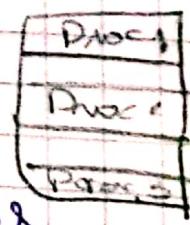
[Proc]

@ hautes

MC



file



## → Tables des bits (Bit Maps) :

C'est sous forme de matrice de 0 et 1

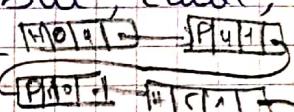
1 : occupé , 0 : vide.

1	0	0	0	1
0	0	0	1	0
0	0	0	1	1
..				

## → Listes chaînées (Linked List) :

Une liste, où chaque élément est construit de :

Etat (H: Libre, P: occupé), @ debut, taille,  
et un pointeur vers l'élément suivant



## Algorithme de placement :

- \* First-Fit : le programme est mis dans le premier bloc qui lui convient
- \* Best-Fit : le programme est mis dans le plus petit bloc qui peut le contenir
- \* Worst-Fit : le programme est mis dans le plus grand bloc mémoire

## Compactage :

On enlève les vides entre les bloc de partition pour avoir un espace mémoire libre plus grand.

## → Allocation non-contiguë

### Pagination :

Découper la RAM en petits blocs de même taille, appelé:

- Dans Mémoire physique: page physique / Frame

- Dans le processus: page .

- A chaque processus, on lui associe une table appelé : Table de pages, tel que :
  - Le nombre de ligne de la table = le nombre de pages du processus.
  - Si la mémoire centrale ne suffit pas, on ajoute une 2ème colonne tel que :
    - colonne 1 : les adresses physiques dans MC.
    - colonne 2 : les adresses physiques dans Swap.
- A chaque page logique (du processus), on associe une page physique (du MC / Swap).

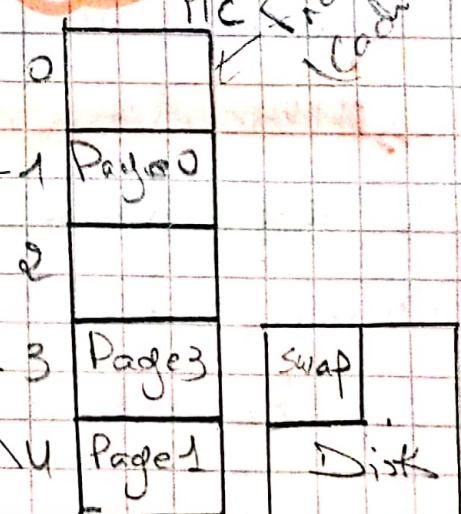
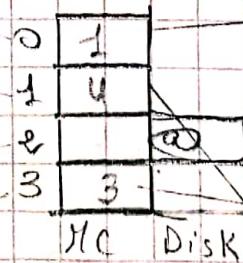
Swap :  
une partie du disque,  
contient des pages comme  
celle de la MC.

Swap in / out :  
des E/S.  
Chaine de référence:  
Les adresses d'un proc

Mémoire virtuelle :

La technique de swap in / out avec une extension  
de mémoire (MC + swap du disque).

Proc	Page 0
1000	Page 1
2000	Page 2
3000	Page 3



- Algorithmes de remplacement :
  - Page victimne
  - Page demandée
- FIFO (First in First out).

On enlève la page la plus ancienne en MC.

- Anomalie de Belady :

En principe : nbr frame  $\uparrow \Rightarrow$  DF  $\downarrow$

Mais Belady a trouvé que : Frame  $\uparrow$  DF  $\uparrow$

- Optimal :

Cet algorithme fait attention à la chaîne de référence aussi ; En effet, il ne fait pas sortir la page qui va être référencé juste après.

Il choisit comme page victimne, la page qui sera référencé la dernière.

- LRU (Least Recently Used) :

Il prend la chaîne de référence, vérifie ce qui a été le plus récemment référencé, et enlève la plus loin référencé.

- Seconde chance (Horloge) :

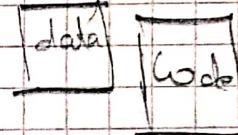
Pour chaque page dans le frame il donne un cpt = 1 initialement, au premier tour pour choisir la page victime, il vérifie le cpt, si il est = 1, il le décrémente et passe au suivant, si il est = 0 la page sera choisie comme page victime.

- Il utilise aussi une tête qui sera déplacé quand il trouve la page victime, ou elle pointera vers la page suivante.

- Si la page existe déjà dans les frames, la tête ne sera pas déplacé, et le cpt de la page sera réinitialisé à 1.

### • Segmentation :

- Chaque unité logique est stocké d'un segment (Bloc mémoire) de taille différente.



- Un programme est constitué d'un :

→ Segment de code.       $\text{@log} = (\text{N}^{\circ}\text{seg}, \text{dep})$

→ Segment de données.       $\text{@phy} = \text{Table Seg} + \text{dep}$

### • Segmentation paginée :

- Il est formé de segments, où chaque segment est paginée.

