Plan de l'exposé 2

III Implantation d'un système

1 Introduction à Unix
2 Structure d'un système
3 Les processus
4 La mémoire

Bibliographie

BOUZEFRANE Samia Les Systèmes d'exploitation : cours et exercices corrigés UNIX, Linux et Windows XP avec C et Java Dunod 2003

Pour une étude avancée :
SILBERSCHATZ,GALVIN,GAGNE Operating System Concepts with Java Wiley 2007

RIFLET La programmation sous Unix Ediscience

Historique :
GOODHEART,COX The Magic Garden Explained. The Internals of Unix system V release 4 Prentice Hall 1994
C.KAISER Systèmes Informatiques Cours B4 Polycopié CNAM
E.GRESSIER Projet CDI Introduction à UNIX

III Implantation d'un système
1 Introduction à Unix

Origines d'Unix Deux principales versions :
1969 Naissance d'Unix Ken Thompson , puis Dennie Ritchie PDP-11 Laboratoires Bell ----> Unix Système V
1978 Université de Berkeley VAX ---> 4.4 BSD
Standardisation de l'interface des appels systèmes Norme POSIX
Unification système OSF/1
1991 Naissance de Linux Linus Thorvalds conçu pour PC et Internet

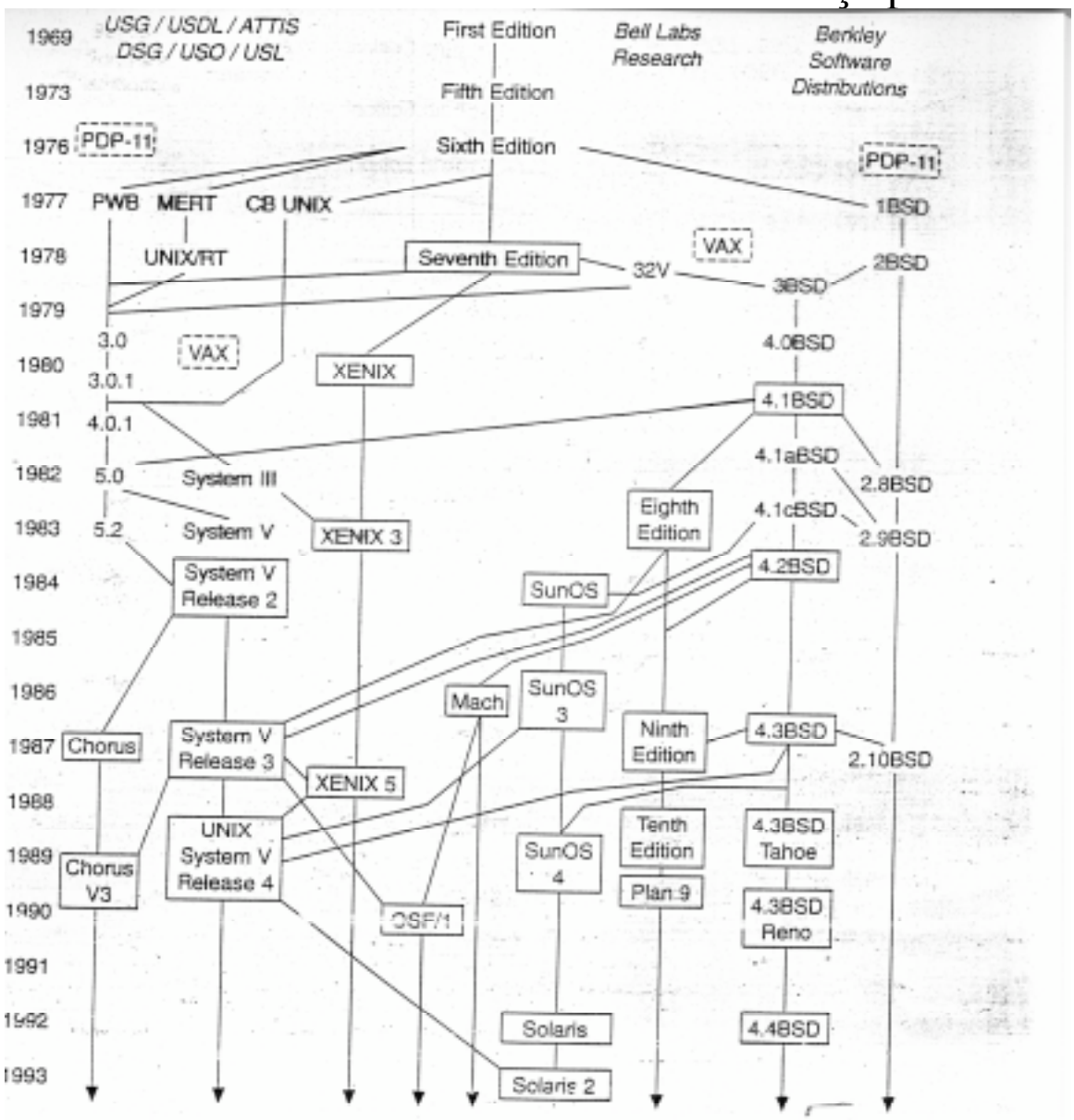


Figure 19.1 History of UNIX versions.

Principes de conception d'Unix

Système conçu par et pour des programmeurs

- Modularité et flexibilité
- Simplicité des algorithmes
- Interactivité et outils d'aide à la programmation
- Portabilité la plupart des programmes sont écrits en C

il suffit de disposer d'un compilateur C pour la machine cible

- Programmation de bas niveau à partir de C

Caractéristiques

Systèmes multiprocessus : multiutilisateurs, multitâches

Organisation basée sur un système de fichiers hiérarchisé : arborescence

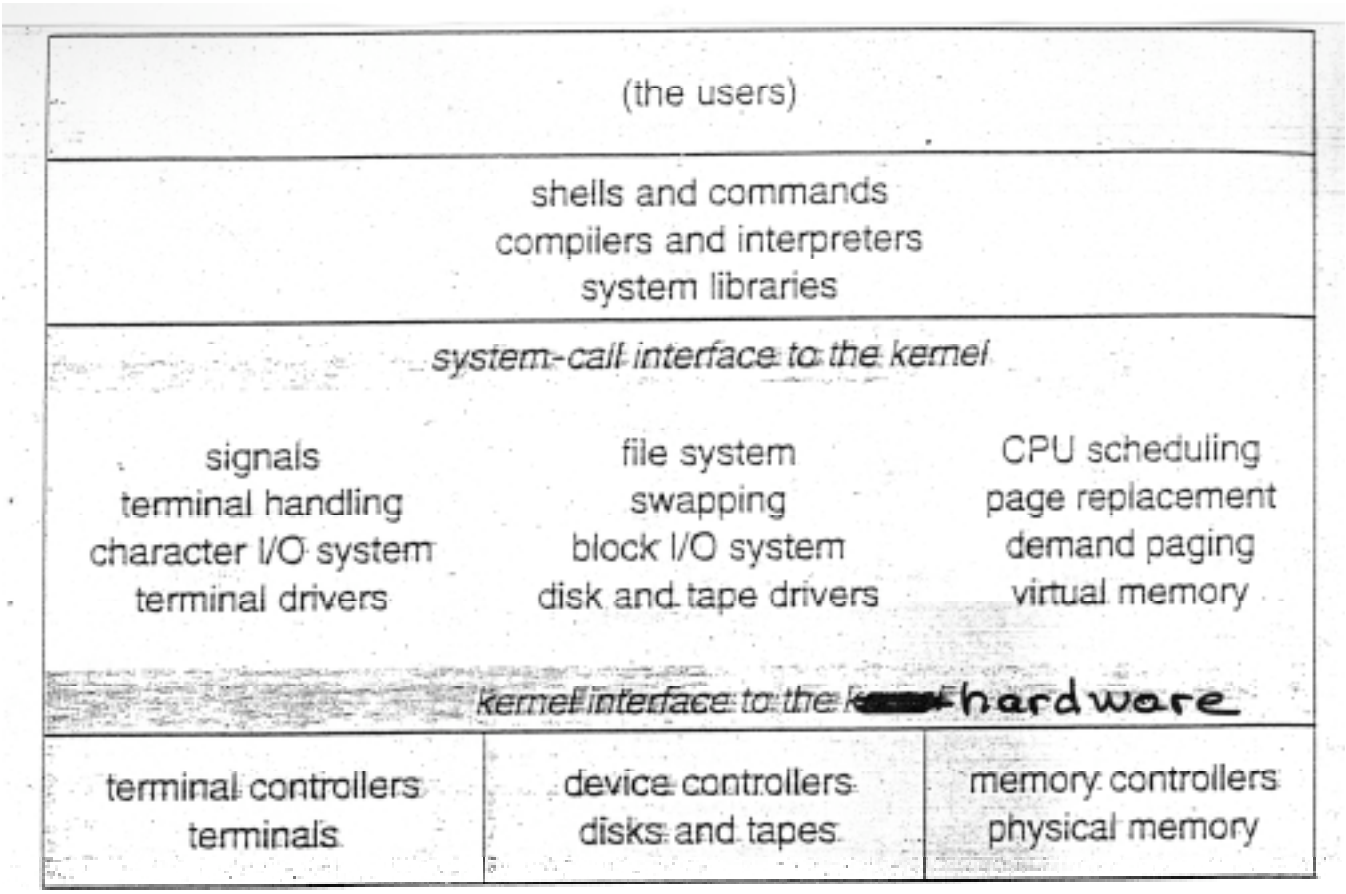
- Environnement d'exécution d'un utilisateur
  sous-arbre (répertoire)
  l'interpréteur du langage de commande (shell) est un processus utilisateur

Machine virtuelle pour l'utilisateur
  Uniformisation des mécanismes d'accès aux objets externes : "Tout est fichier" :
    programmes, données des utilisateurs
    programmes systèmes
    périphériques

2 Structure d'un système

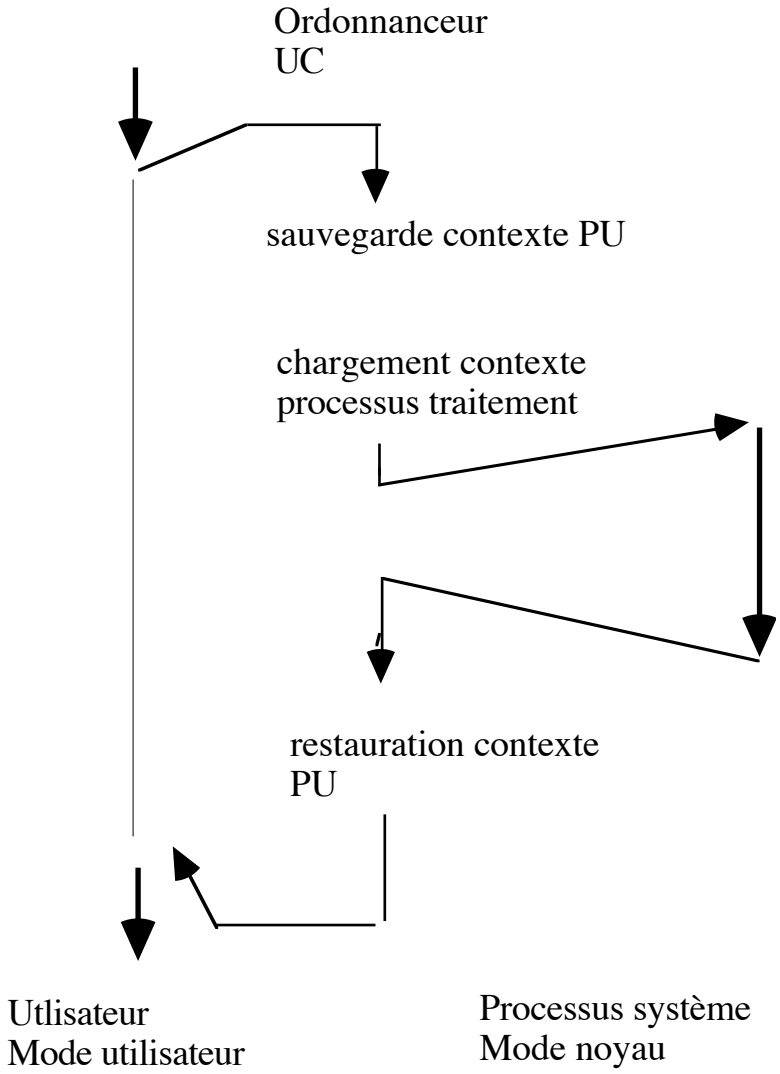
Exemple d'Unix

Système multiprogrammé et temps partagé



Interface de programmation

Appels système



Types d'appels systèmes

Manipulation de fichiers

accès par un chemin dans l'arborescence des répertoires Unix
- périphériques /etc/ttys
- programmes systèmes /usr/bin
- fichiers utilisateurs /usr/...

appels : create, open, read, write, close etc...

Gestion des processus

fork, exec, exit , kill, wait, sleep,wake-up...

Signaux

conditions exceptionnelles --> interruption logicielle
référence mémoire invalide, alarme, instruction machine illégale
exemple ctrl Z --->SIGINT suspension du processus en cours

3 Les processus

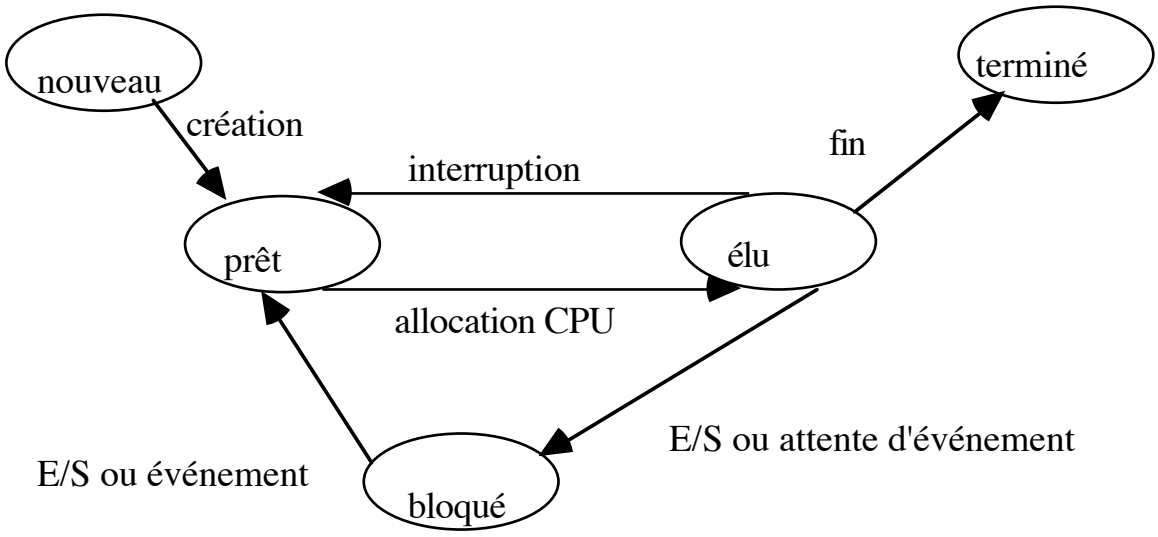
Processus

Programme en exécution
code, données + mot d'état programme et contexte d'exécution

Le mot d'état programme (PSW) : état de la machine
- compteur du programme (Compteur Ordinal)
- registres de l'UC

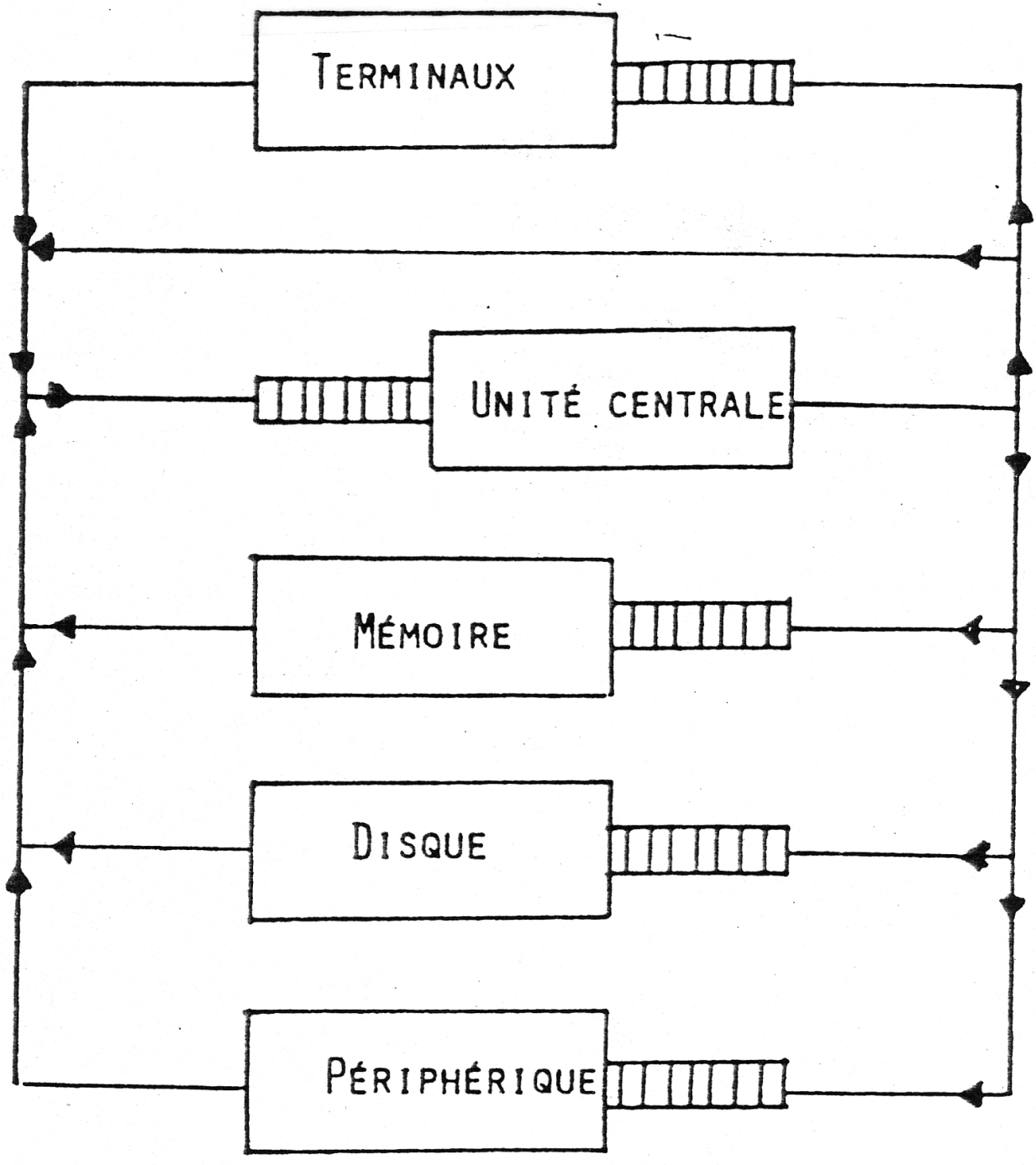
Le contexte d'exécution
- statut d'accès à l'UC (priorité)
- information sur la mémoire allouée au processus
- informations statistiques
- Informations d'E/S (fichiers,périphériques alloués)
- Etc…

états d'un processus en multiprogrammation



Ordonnancement des processus

Files d'attente + ordonnanceur pour chaque ressource



MODELE D'UN SYSTEME EN TEMPS PARTAGE

Opérations sur les processus

Création
Terminaison normale ou erreur
Suspension,réveil

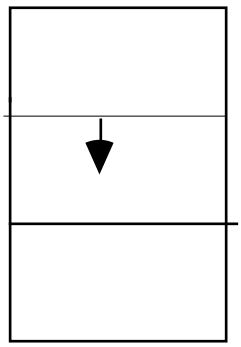
Coopération entre processus

Processus concurrents : indépendants ou coopérants
- partage d'information exemple fichier partagé
- synchronisation : division d'un programme en plusieurs tâches concurren

Communication entre processus

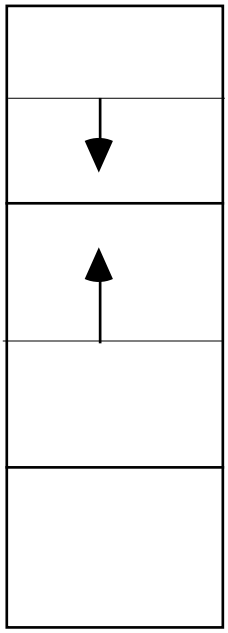
partage de mémoire
messages

Cas d'Unix
Représentation d'un processus en mémoire



pile noyau

contexte utilisateur



pile utilisateur

données utilisateur

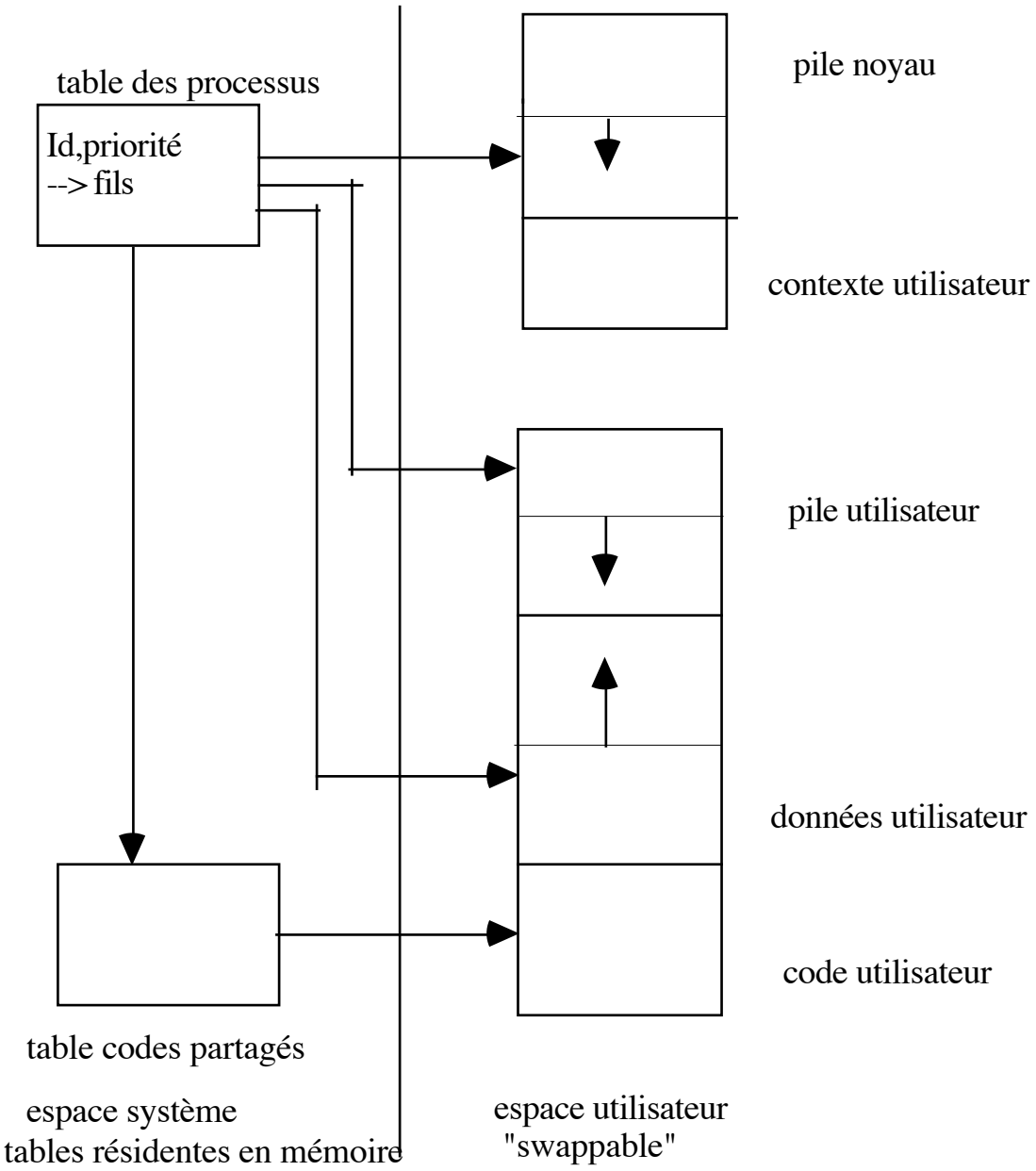
code utilisateur

pile noyau exécution de programmes systèmes pour l'utilisateur
lors d'appels système

contexte utilisateur
contexte UC (registres généraux, pointeur de pile,
pointeur table des pages)
environnement
table des pages
table des processus

pile utilisateur
variables temporaires, adresse retour de procédures ...

Vue d'un processus par le système



Création de processus

1 utilisateur = plusieurs processus

appel système **fork**
crée une copie du processus appelant : contexte, pile, code, données
un lien dans la table des processus de l'utilisateur

Remarques

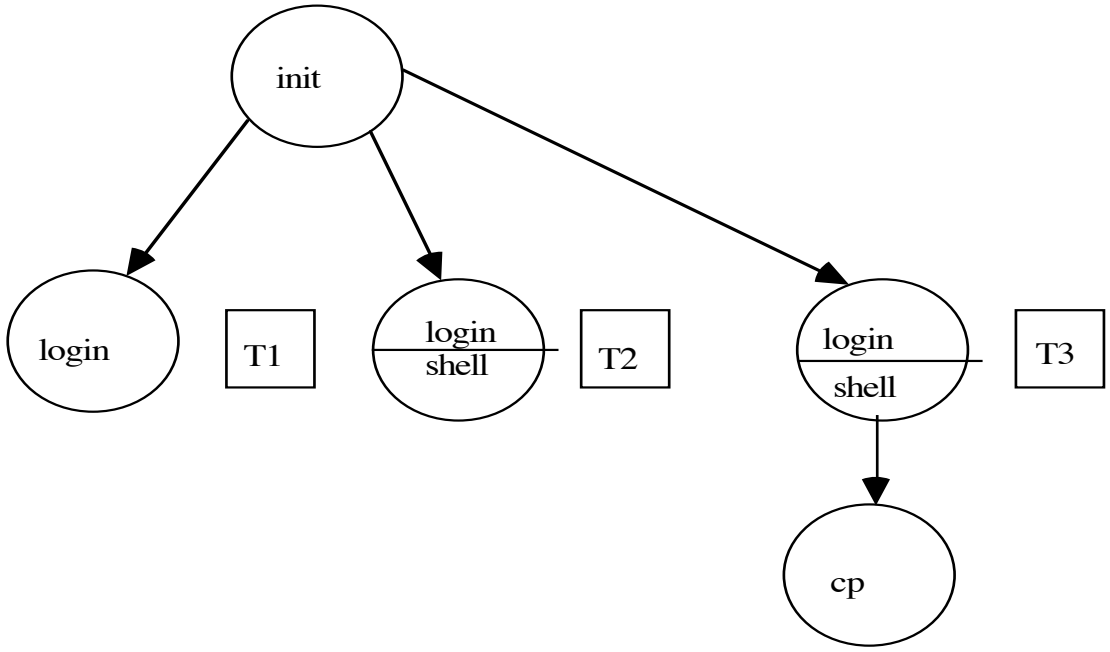
- En fait, le code n'est pas dupliqué mais partagé (réentrance des programmes), les données ne sont copiées que si elles sont modifiées par le père ou par le fils, le fils aura toujours initialement la vue des données lors du fork
- Les fichiers du père ouverts au moment du fork sont partagés entre le père et le fils

Comment savoir si un processus exécute le code du père ou du fils ?
fork renvoie un identifiant de processus `pid`

```
pid=fork(); si fork réussit pid>=0
si pid<0 alors échec
sinon si pid>0 alors code du père
sinon code du fils
```

- un processus peut avoir un arbre de descendants

Exemple : gestion de terminaux

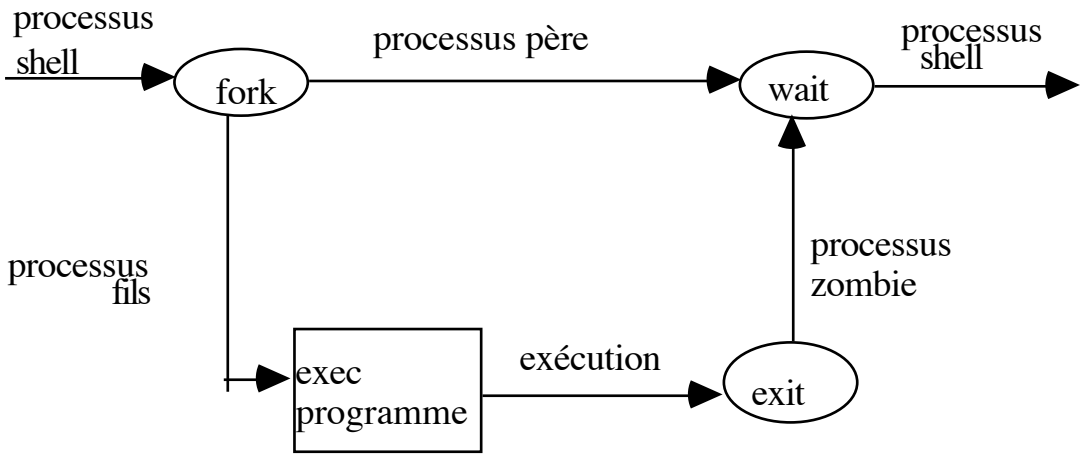



```
A l'inialisation, init est activé -- lecture
/etc/ttys
Création d'un fils pour chaque terminal;
mise en veille de init jusqu'à ce qu'un fils se
termine
exécution des fils
   programme login
     création du processus shell
     tant que non terminé faire
     attente d'une commande
       création du processus commande
       exécution de commande
```

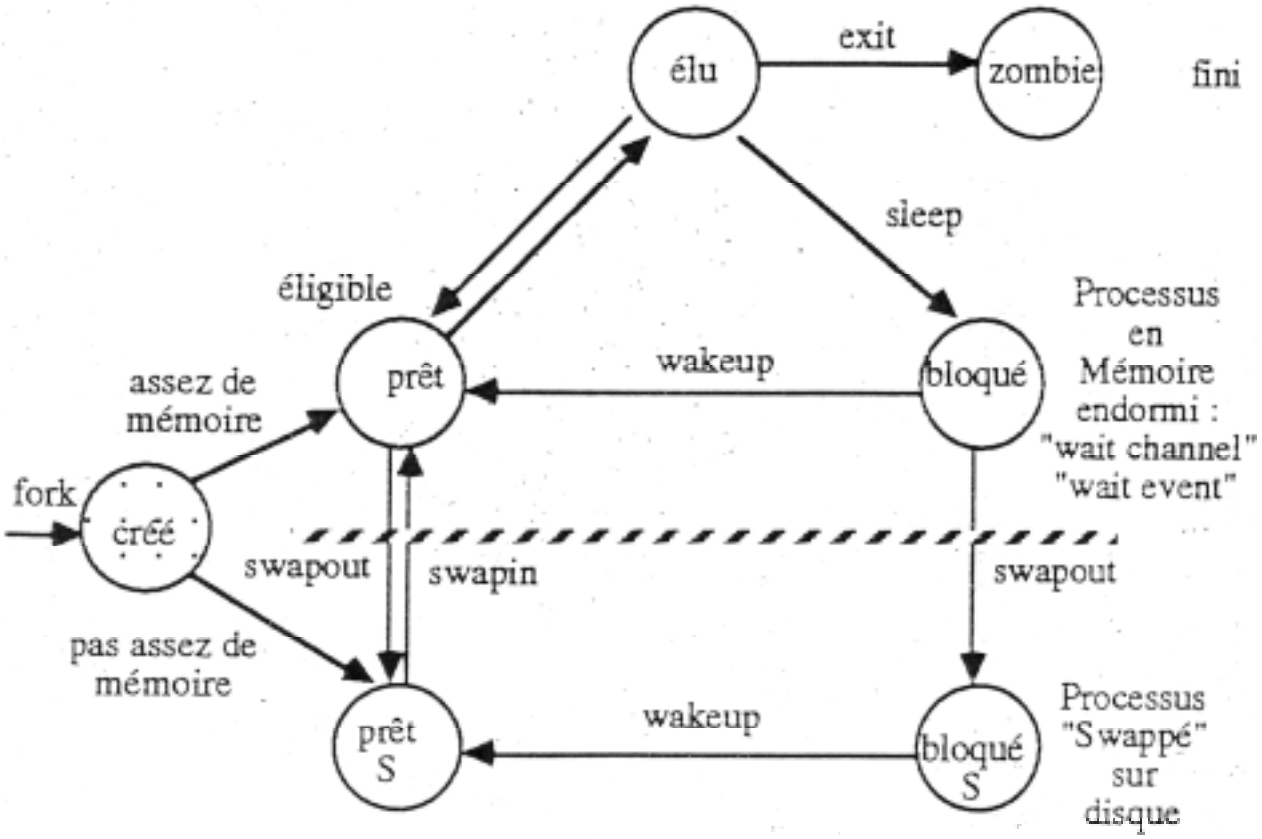
La commande ps permet de connaître les processus d'un utilisateur et les processus système s'exécutant pour lui

Création d'un processus pour exécuter un programme

création appel système **fork()**
exécution des commandes : appel système **exec()**
charge un nouvel exécutable en mémoire
terminaison d'un processus **exit()**
attente de la terminaison d'un fils **wait()**



Les états d'un processus



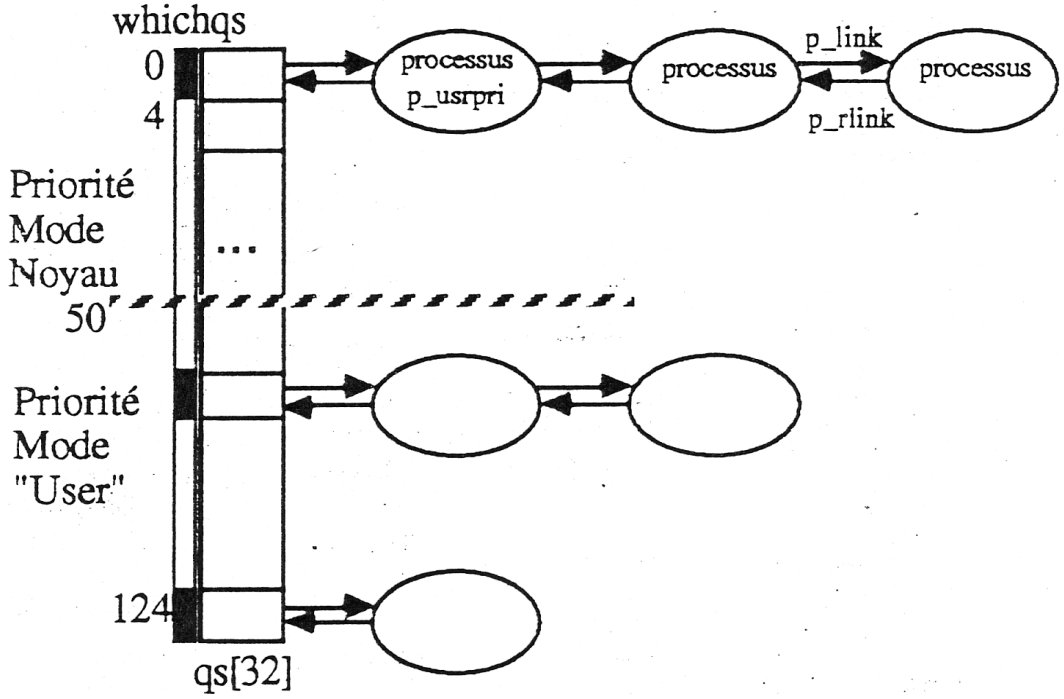
Ordonnancement du processeur

- temps partagé : allocation par *quantum* de temps (0.1s) aux processus

- ordonnancement selon priorités : à chaque est associée une *priorité*
processus du noyau plus forte priorité (p<0), non préemptible
processus utilisateurs plus faible priorité (p>0), préemptible,
périodiquement (1 s) réévaluation des priorités

- File d'attente des processus prêts
tableau de files par priorité
recyclage dans la file en fin de quantum (tourniquet)

File des Processus Prêts -> Tableau de files, sauf le processus élu



- Activation de l'ordonnanceur
interruption horloge
demande d'E/S
fin d'E/S
attente d'un événement

4 La mémoire

Espaces d'adressage d'un programme

Espace d'adressage physique : ensemble des adresses mémoire centrale des instructions et des objets que le programme référence

Espace d'adressage logique (virtuel)
En général, le programme est enregistré sur disque comme un fichier binaire exécutable, l'ensemble d'adresses mémoire ne prenant pas en compte l'implantation du programme en mémoire.

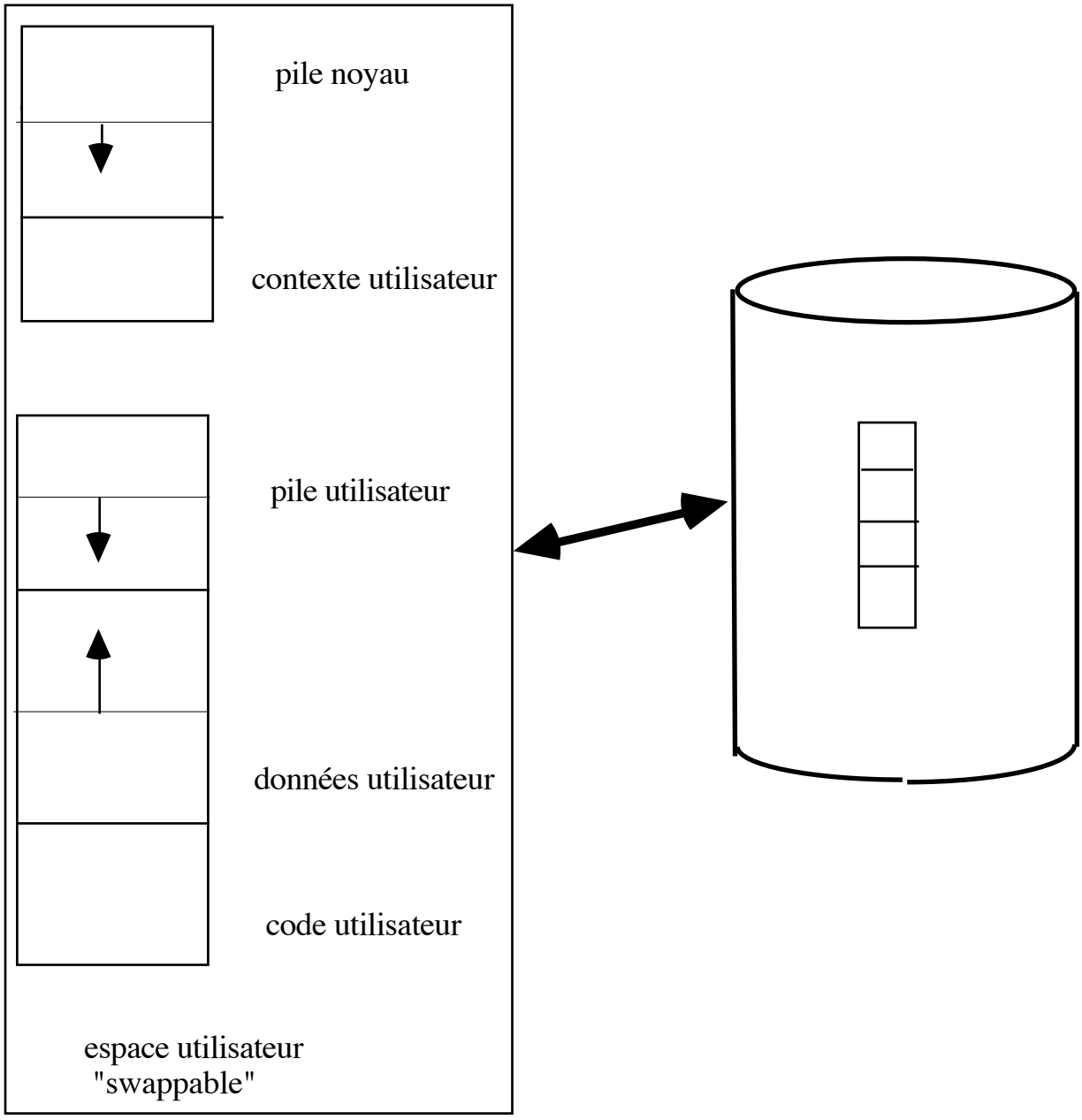
Correspondance entre les espaces d' adressage logique et physique

Quand ?
- à la compilation :
adresse d'implantation en mémoire connue
code absolu adresses du programme=adresses mémoire
Inconvénient si l'adresse d'implantation change, il faut recompiler
- au chargement :
code relatif (relogeable, translatable)
- à l'exécution :
programme peut changer de place en mémoire en cours d'exécution.
Un dispositif matériel spécifique (Memory Management Unit) transforme une adresse logique en une adresse physique au moment de l'exécution.

Va-et-vient (swapping)

Principe

Manque de mémoire => transfert de l'image mémoire du processus sur disque Va (swapping out)
Assez de mémoire =>chargement du processus Vient (swapping-in)



Fonctionnement

- manque de mémoire
allocation dynamique de mémoire : processus augmente son espace mémoire au cours de son exécution, l'espace libre diminue au détriment d'autres processus

- le choix d'un processus : réalisé par le"swapper"(ordonnancement des processus pour la mémoire)
périodiquement (4s) réévaluation des processus à retirer et à charger
critères de choix :
retrait :
inactivité d'un processus depuis longtemps,
taille du programme, ancienneté de chargement
chargement ;
swappé depuis longtemps, petit programme

- les actions du système
retrait
mise à jour du descripteur de processus, libération de la mémoire, libération des ressources allloués au processus
problèmes : entrées sorties pendantes, attente d'un événement
chargement
mise à jour du descripteur, allocation de mémoire, allocation d'autres ressources

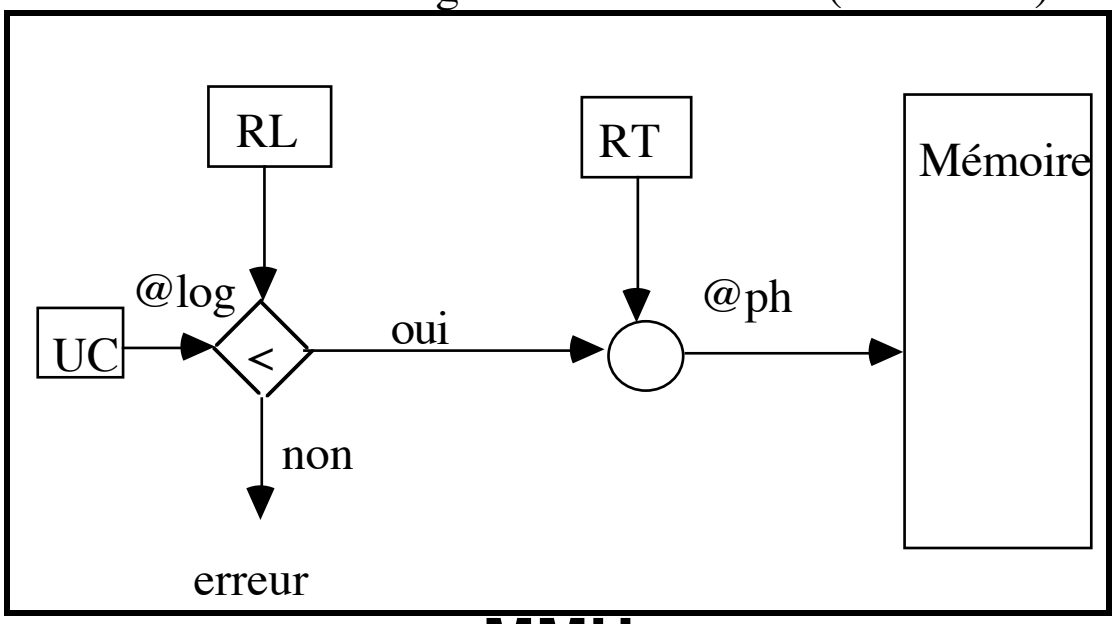
Remarques

Le swapping n'est en général utilisé qu'en dernier ressort dans les systèmes Unix (Version Berkeley) coût de transfert d'un processus en mémoire; la gestion est en amont par le mécanisme de pagination

Allocation contigüe

Partition unique

Monoprogrammation
protection du moniteur : Registre de limite
translation d'adresse : Registre de translation (relocation)



Problème : insuffisance mémoire pour le processus
->recouvrements

Partitions multiples

Multiprogrammation : plusieurs programmes en mémoire
- Protection du système
- Protection entre utilisateurs
- Registre de translation pour chaque programme

Ordonnanceur à long terme

Les programmes sont stockés sur disque :
- Choix dans la file d'attente d'un programme à exécuter

Les problèmes
- Choix d'une zone de mémoire adéquate (first fit,best fit, worst fit)
- La fragmentation interne (toute la zone allouée n'est pas utilisée)
- La fragmentation externe : les allocations successives créent des trous ;
- Solution : compactage mais difficile et coûteux

La pagination

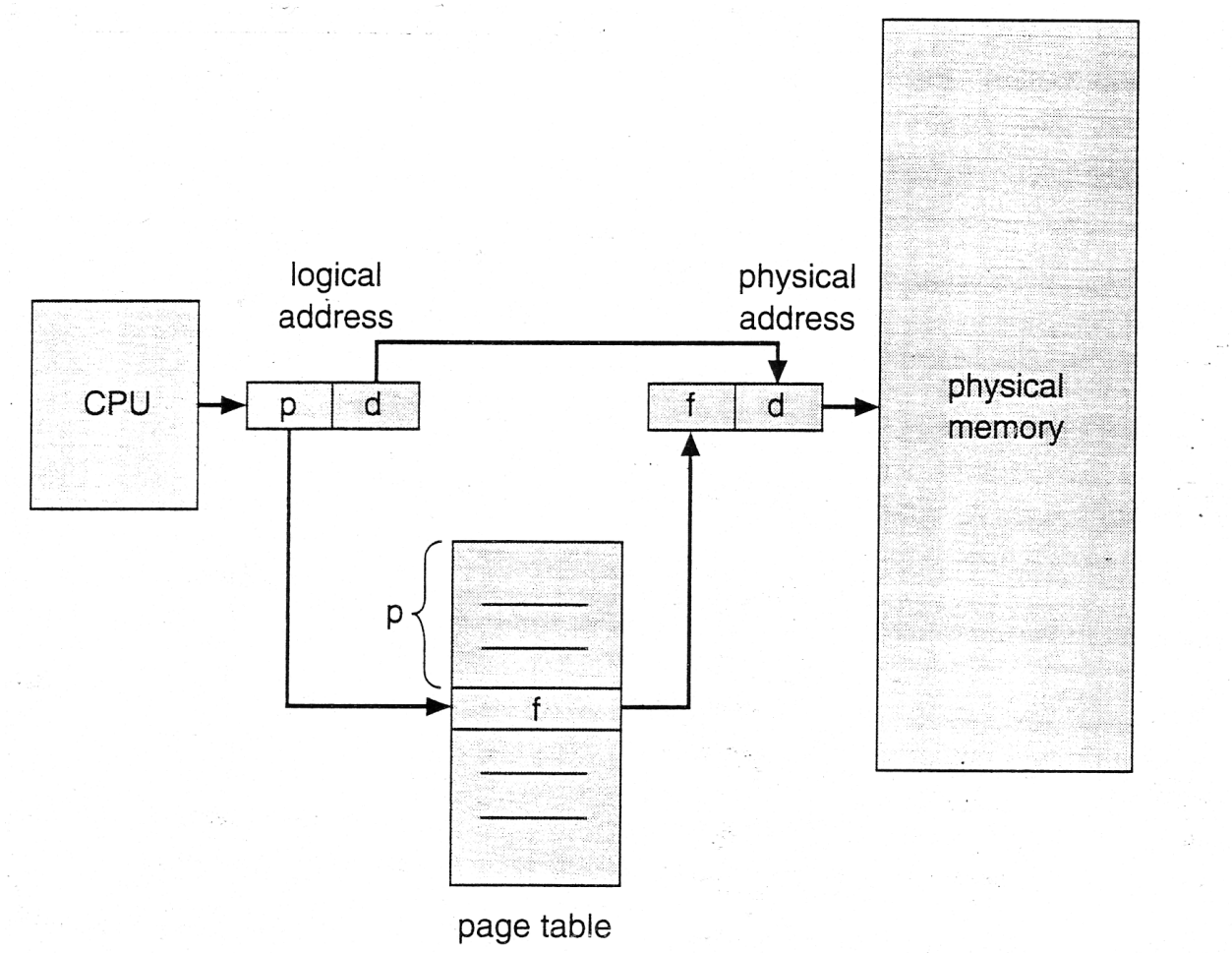
Principe

Allocation par blocs de taille fixe :
Mémoire physique = suite de blocs de taille fixe : *cases*
Mémoire logique = suite de blocs de taille fixe : *pages*
Taille case = taille page

Correspondance par le MMU(Memory Management Unit)

Adresse logique

n° de page	déplacement



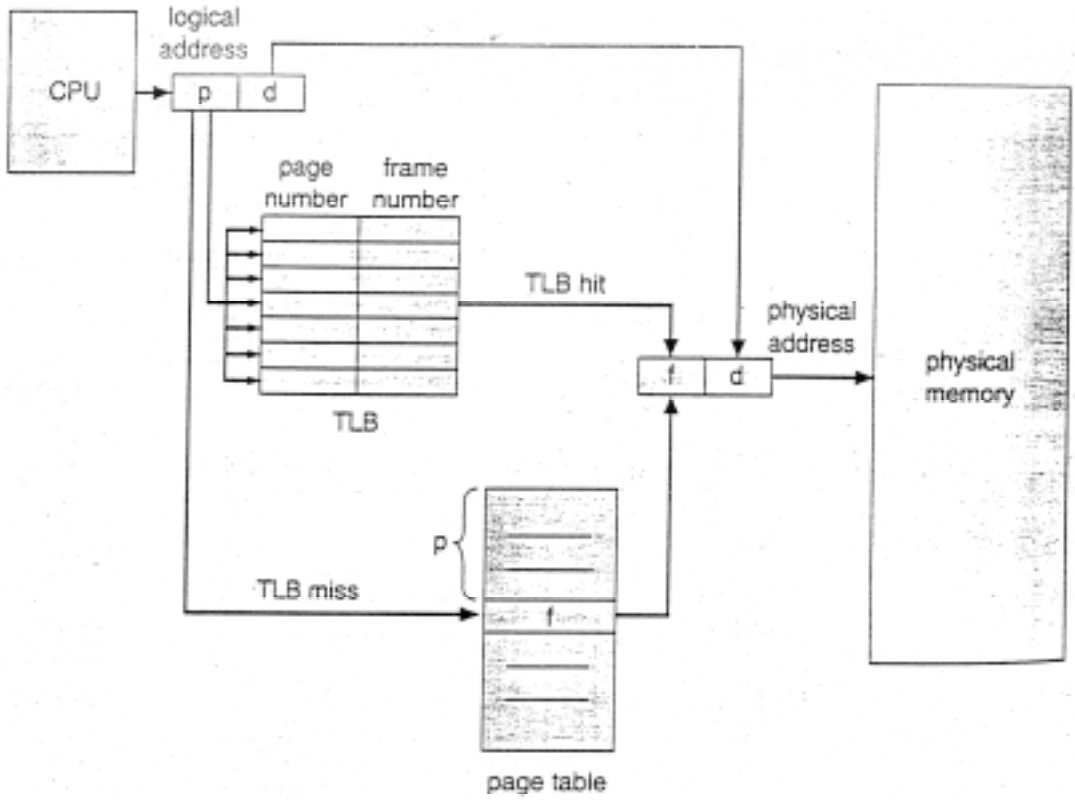


Figure 8.16 Paging hardware with TLB.

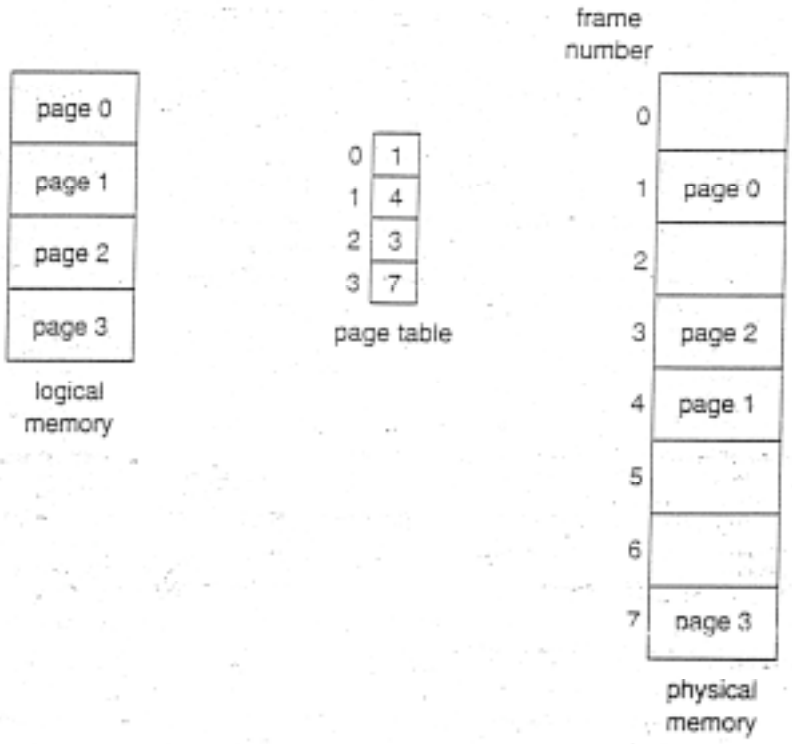


Figure 8.13 Paging model of logical and physical memory.

Mémoire virtuelle

Objectif : offrir à l'utilisateur un espace d'adressage supérieur à la place mémoire allouée
Principe de localité
un processus exécute pendant un certain temps une partie du code (localité temporelle) et utilise une partie des données (localité spatiale).

Idée ne maintenir en mémoire que le code et les données utiles,

l'ensemble du programme étant conservé sur disque de pagination
--> allocation d'espace disque à la création des processus

Mémoire virtuelle paginée

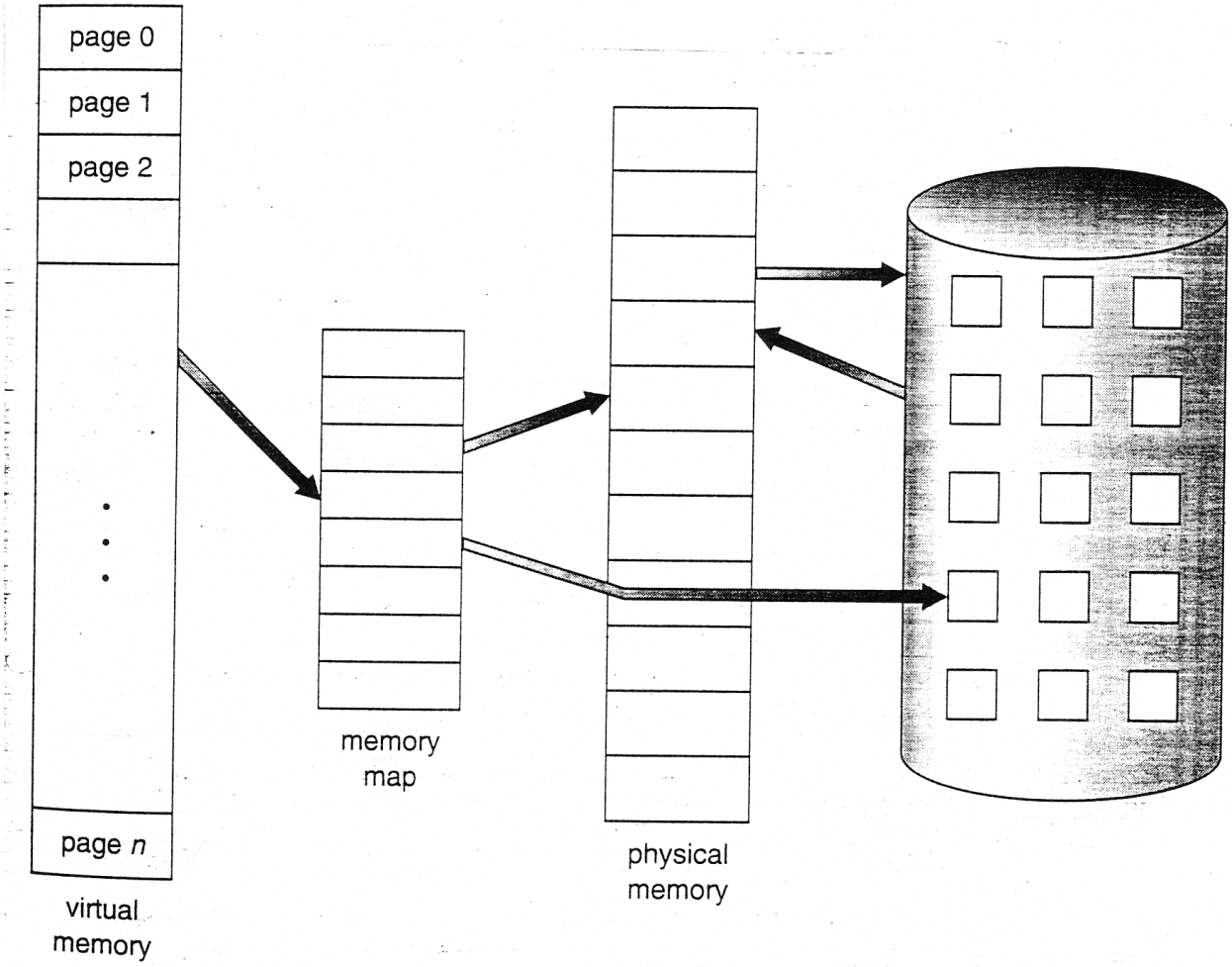


Figure 9.1 Diagram showing virtual memory larger than physical memory.

Table des pages d'un processus

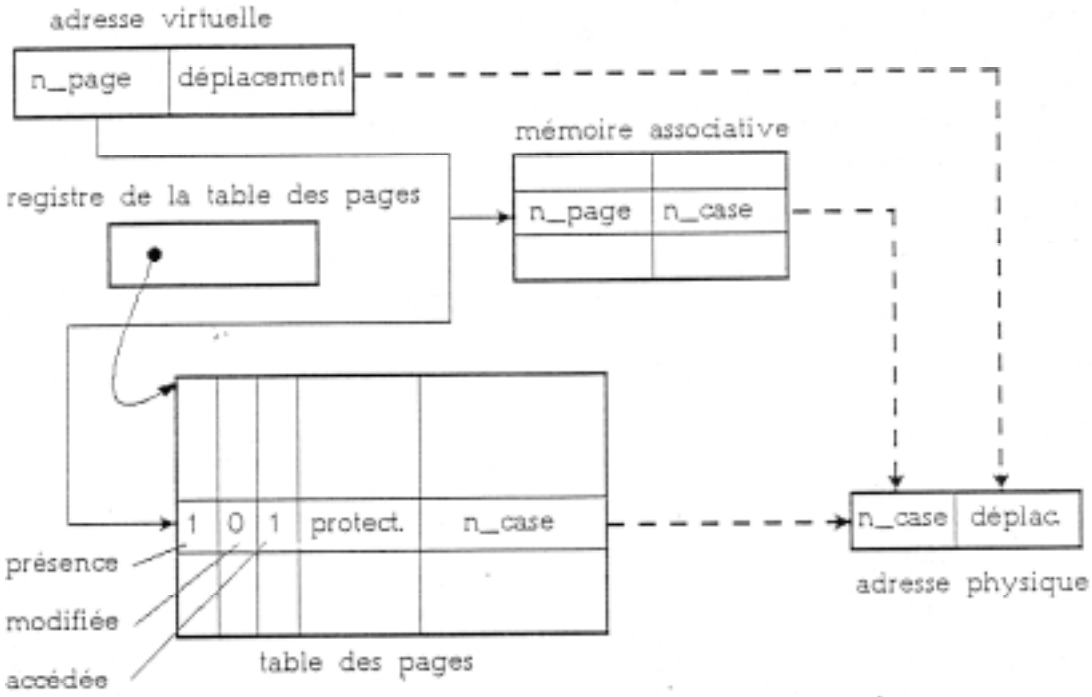


Fig. 14.5. Fonctionnement de la pagination à un niveau.

Fonctionnement

MMU

Recherche d'une instruction ou d'un opérande en mémoire
si page présente en mémoire alors
adresse= n°case + déplacement
sinon défaut de page

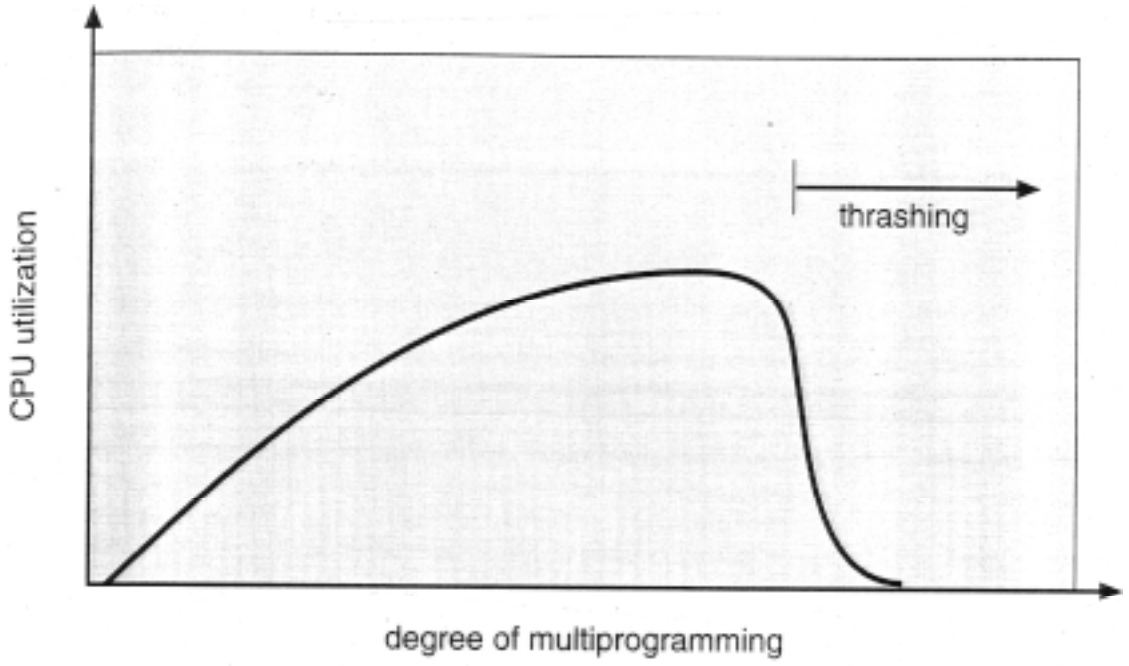
Défaut de page=>appel système gestion des défauts de pages
- si case libre alors charger page dans la case
- sinon libérer une case : choisir une case occupée;
transférer la page correspondante sur disque;
charger la page en défaut dans la case;

Choix d'une case => algorithme de remplacement de pages
par exemple choisir la page la plus anciennement référencée
(LRU)

Ecroulement du système

Problème :
beaucoup de processus en mémoire =>peu de place mémoire par processu
=>nombreux défauts de pages => file d'attente au disque
augmente=>ralentissement des processus

écroulement du système



Remèdes
Abaisser le taux de multiprogrammation
Unix utilise le va-et-vient par exemple