Systèmes d'exploitations

- angl. « Operating System (OS) »
- Qu'est-ce que c'est?

  « Programme assurant la gestion de l'ordinateur et de ses périphériques »

- A quoi ca sert?
  - à simplifier la vie des utilisateurs et des programmeurs
  - à gérer les ressources de la machine d'une manière efficace

Abstraction

- Cacher la complexité des machines pour l'utilisateur afin d'utiliser la machine sans savoir ce qui est derrière

- Abstraction du terme « Machine » selon Coy:
  - machine réelle = Unité centrale + périphériques
  - machine abstraite = machine réelle + système d'exploitation
  - machine utilisable = machine abstraite + application

Exigences à un Système d'exploitation

- Généralités
  - Satisfaire les utilisateurs et les programmeurs
  - Gérer 2D, 3D, vidéo, audio, réseau, CD, DVD, clé USB, ...
  - Plusieurs utilisateurs (itinérants) --> multi-utilisateurs
  - être extensible

- De plus en plus gros et complexe :
  - Efficace, évolutif, maintenable

Exigences de l'utilisateur

- « Faut que ça marche ! »
  (comme j'en ai envie ...)
- « Ça imprime pas ... »
- = Machine utilisable (machine étendu)

Exigences du programmeur

- Simplifier l'accès aux ressources de la machine :
  - Mémoire, processeur, périphériques, fichiers, programmes, réseaux, communication interne
  - Modèle de programmation simple et unifié
- Efficacité dans tous les cas
- = Machine étendue

Quelques définitions

- Processus
- Traitement par lots
- Systèmes Multi-tache
- Systèmes Multi-utilisateurs
- Systèmes Multi-processeurs
- Systèmes temps réel
- Systèmes distribués

Définitions: Processus

Déf.:

Un processus est un programme lors de l'éxécution

(aspect dynamique d'un programme)

Définitions: Traitement par lots (Batch processing)

- Un utilisateurs donne plusieurs commandes (« Jobs ») dans une queue d'éxécution de programmes
- Entièrement séquentielle
- p.ex. pour faire plusieurs calculs pendant la nuit

- p.ex. **`autoexec.bat`**

Définitions: Systèmes Multi-tache (Multitasking)

- Assurer l'exécution de **plusieurs programmes** en **meme temps (c-à-d. plusieurs processus)**

- Chaque processus a besoin du processeur
  - situation concurrente
  - solution: « scheduling »

Définitions: Systèmes Multi-processeurs

- système avec plusieurs processeurs
  - parallèle
  - vrai multi-tache
  - doit assurer qu'il y a l'éxecution d'autant de processus que processeurs en meme temps
- contrairement: système avec un seul processeur
  - quasi-parallèle
  - arreter et reprendre les différentes processus
    - Gestion avec le « scheduler » (ordonnancement des processus)

Définitions: Systèmes Multi-utilisateurs (« time-sharing »)

- permettre à **différentes personnes** de travailler avec **un ordinateur** en **même temps**
- connexion par
  - via le terminal de l'ordinateur lui-même
  - à distance (telnet, ssh, ftp, ...)
- donner l'impression à chaque utilisateur qu'il est seul
- exige une gestion des droits
  - de fichiers (pour éviter la destruction des fichiers etc.)
  - de processus

Définitions: Multi-utilisateurs

- Login
- Type:
  - Administrateur (« root »)
  - Groupes
  - Utilisateurs
- pour gérer les droits

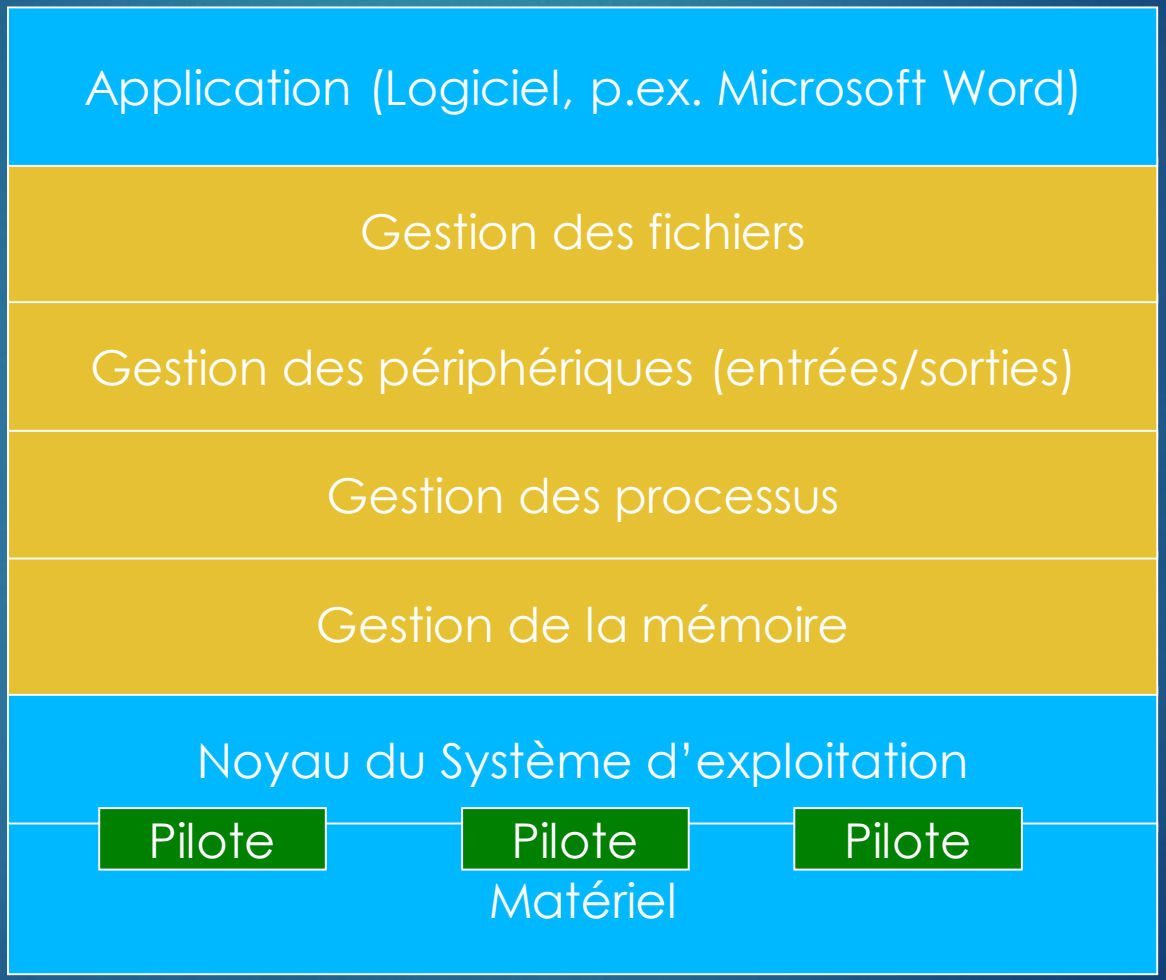
Définitions: Systèmes Temps réels

- Sert pour le pilotage et le contrôle des déroulements externes
- doit garantir des temps de réactions données pour des signaux extérieur urgents
- plusieurs systèmes d'exploitations n'y arrivent pas car l'interruption de certaines activités met le système dans un état instable

Définitions: Systèmes distribués

- doit permettre l'éxecution d'un **seul programme** sur **plusieurs machines**
- distribuer les processus et les remettre ensemble
- pour gros calculs, p.ex. inversion de grandes matrices

SE: Modèle en couches



Gestions

- Gestion de la mémoire
- Gestion des fichiers
- Gestion des processus
- Gestion des périphériques (entrées/sorties)
  - Contrôle des péripheriques via « Pilotes » (Driver)
- Quelques logiciels
  - Logiciels utilitaires (ls, pwd, format, ...)
  - Logiciels d'application (Bloc-notes, ...)
  - Logiciels de communication (Internet Explorer, ...)

Systèmes d'exploitations

- CP/M (depuis 1974), Digital Research
- UNIX (depuis 1969-1979), premier par AT&T
- MS-DOS (depuis 1981), Microsoft
- MacOS (depuis 1984), Apple
- Windows (depuis 1991), Microsoft
- Linux (depuis 1992), OpenSource

Systèmes d'exploitations

- CP/M (depuis 1974), Digital Research
  - Gestion de disque dur, mais pas d'arborescence
  - Pas de graphisme
  - Exemple:
    - CPU 8088, 2 MHz
    - 64 KO de RAM
    - 5 MO de disque dur
  - cf. la loi de Murphy

Systèmes d'exploitations

- UNIX (depuis 1969-1979), AT&T
  - a servi de modèle pour MS-DOS, Windows, ..
  - Multi-tâche et Multi-utilisateurs
    - accès simultané aux fichiers, péripheriques, mémoire, processeurs, ..
  - Protection mémoire : aucun programme ne peut faire planter le système
  - systèmes de fichiers hiérarchique
  - GUI X-Windows

Systèmes d'exploitations

- MS-DOS (depuis 1981), Microsoft

Systèmes d'exploitations

- MacOS (depuis 1984), Apple
  - premier GUI

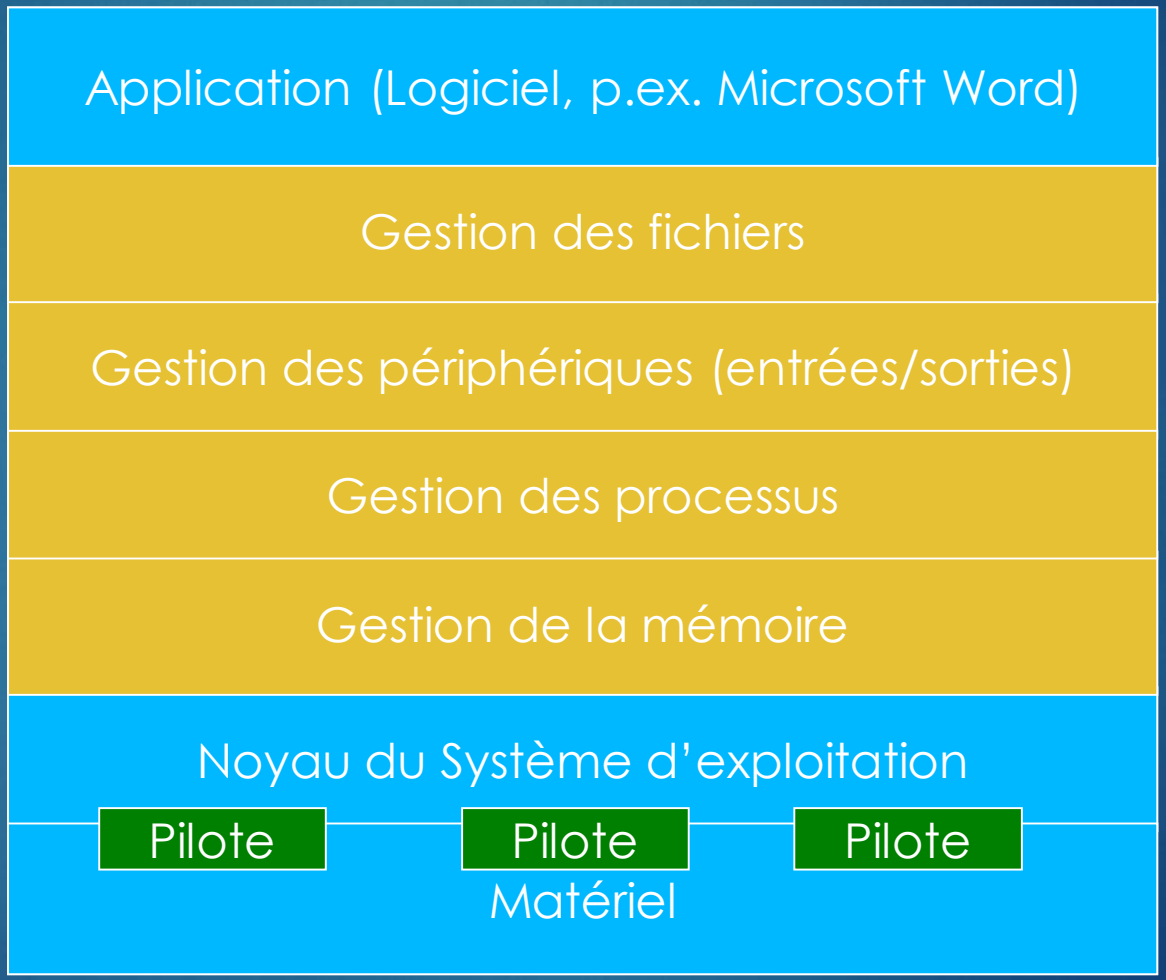
Systèmes d'exploitation Windows

- Windows 3.11
  - pas de multitâche, pas de multi-utilisateurs
- Windows 95
  - **multi-tâche**
  - premier système 32 bit
- Windows 98
  - Internet integré dans le GUI
  - Plug & Play
- parallèlement Windows NT
  - système d'exploitation réseaux **multi-utilisateur**
- Windows 2000, et après Windows XP
  - jumellage entre système d'exploitations réseaux et « stand-alone »

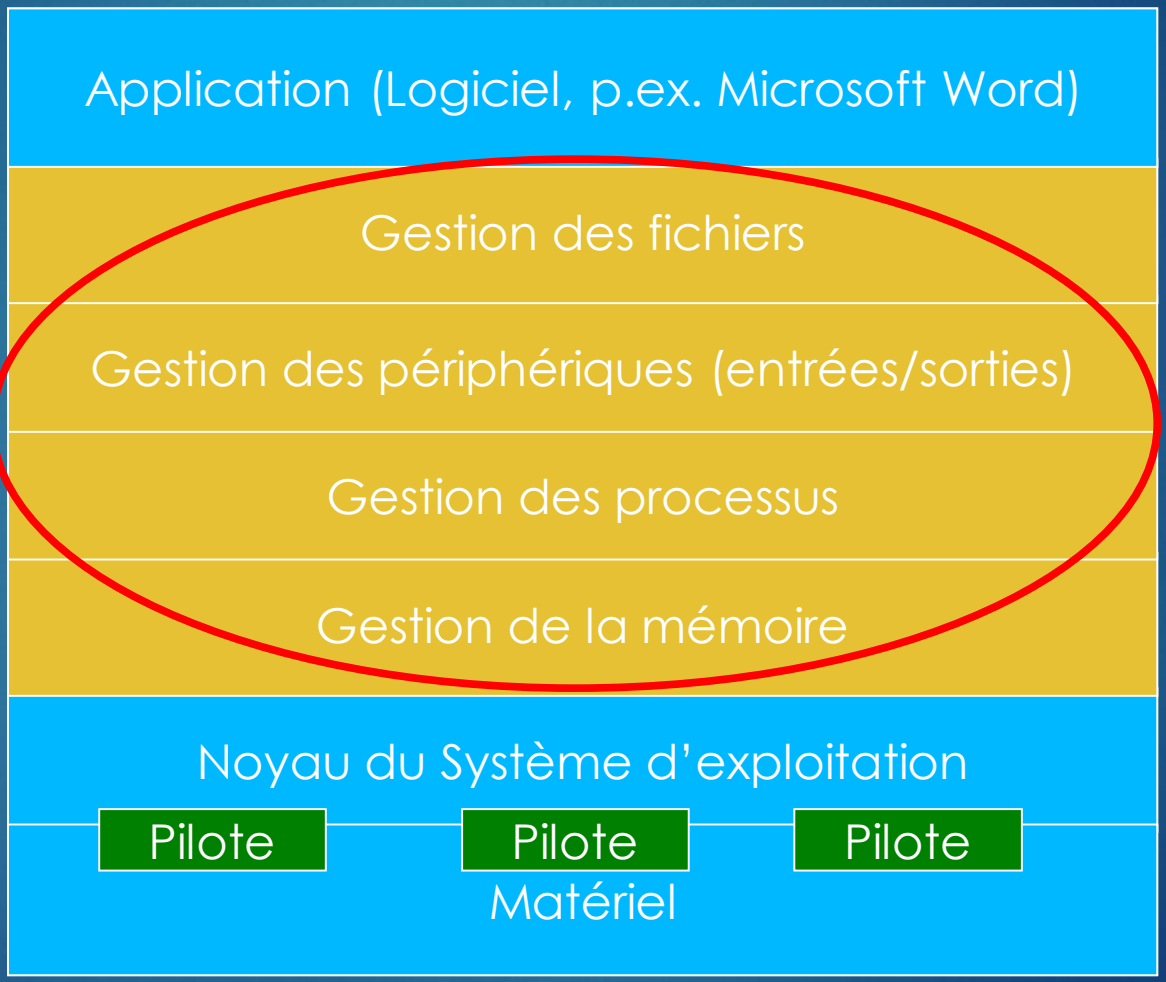
Systèmes d'exploitations

- Linux (depuis 1992), OpenSource
  - finlandais Linus Thorwald
  - Licence GPL (General Public Licence) – OpenSource
  - Multi-tâche et Multi-utilisateurs
  - Distributions
    - Red Hat
    - Fedore
    - S.u.S.e
    - Debian
    - Mandrake..

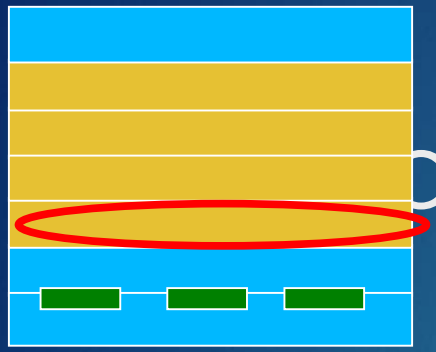
Modèle en couches



Modèle en couches



Modèle en couches



on mémoire (1/6)

Pour illustrer le travail du système d'exploitation, on choisit de décrire quelques tactiques appliquées par les SE pour gérer la mémoire :

1. les partitions,
2. le tassage,
3. la pagination,
4. la mémoire virtuelle.

Pour chacune on présente l'idée, la mise en œuvre, les avantages et les inconvénients.

on mémoire (2/6)

les partitions

- Technique « historique ».
- Idée : Diviser arbitrairement la mémoire en **partitions** de dimensions fixes : partitions statiques. Toutes les partitions ne sont pas de même taille.
- Mise en œuvre : Les programmes sont implantés dans chaque partition.

- Avantages : Simplicité du SWAP
- Inconvénients : Un gros programme ne peut être exploité que dans une partition suffisamment grande et un petit programme rentabilise mal une grande partition.

on mémoire (3/6)

le tassage

- Idée : Banaliser la mémoire et tasser les programmes les uns à la suite des autres.
- Mise en œuvre : …
- Avantages : la mémoire centrale est mieux utilisée. A un instant donné, il ne peut y avoir qu'une partition de libre.
- Inconvénients : le tassage de la mémoire est coûteux en temps de traitement.

on mémoire (4/6)

La pagination

- Idée : Découper la mémoire en PAGES de dimension fixes et implanter les programmes dans les pages libres.
- Mise en œuvre : La conséquence est un éparpillement des programmes en mémoire. Cela nécessite donc sous forme matérielle (ou logicielle) la mise en place d'une table d'occupation des pages, qui reconstitue l'ordre logique des différentes parties des programmes.
- Avantages : Ce type de gestion est très efficace car utilise au maximum la mémoire et évite le tassage de mémoire.
- Inconvénients : on reste toujours limité à la taille effective de la mémoire.

on mémoire (5/6)

La mémoire virtuelle

- Idée : Supprimer la contrainte de **dimension** de la mémoire.
- Mise en œuvre : On dote l'ordinateur de deux niveaux de mémoire : la mémoire centrale (performante et chère) et la mémoire de masse (le disque) moins performante mais de dimension pratiquement illimitée.
  - Performances liées aux : seek time ; latency time ; transmission time. Là encore l'optimisation est gérée par le système d'exploitation qui met en place des stratégies.

n mémoire (6/6)

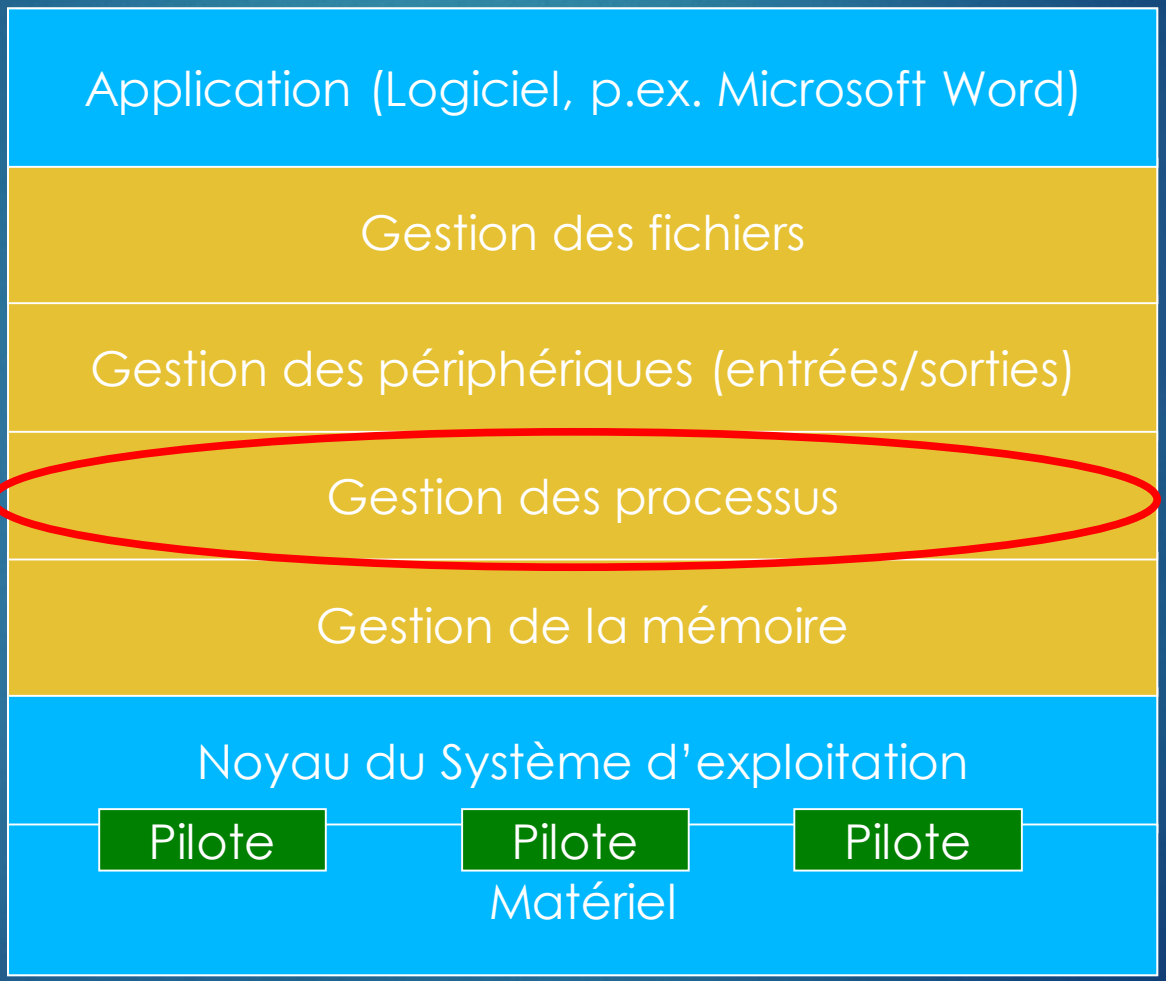
La mémoire virtuelle (suite)

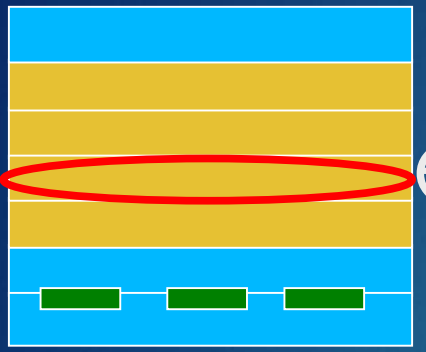
- Enfin, est mis en place une stratégie (algorithme) de choix de la page dont dépend beaucoup la rapidité du système.
- Avantages : Beaucoup de place !
- Inconvénients : SWAP ou accès disque à optimiser

Remarque 1 : Vient en combinaison d'autres techniques de gestion de la mémoire.

Remarque 2 : d'autres stratégies ou combinaisons de stratégie existent … !

Modèle en couches

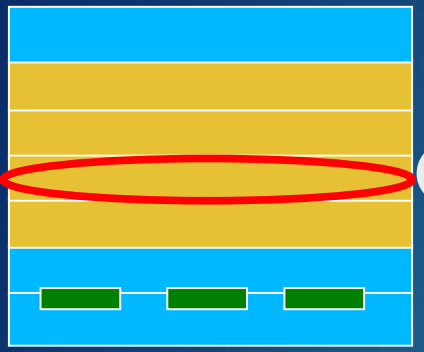




essus (1/5)

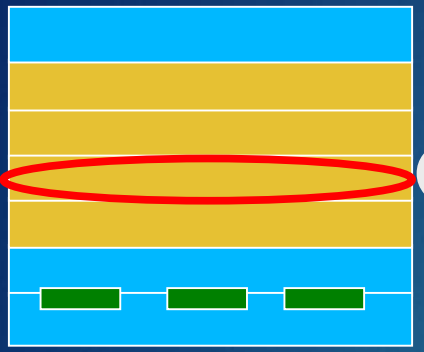
Cette partie s’appuie sur le document *"Ecrire des applications réseau sous Linux", 3ième partie, Alain Basty, Linux Magazine, octobre 1999*" qui présente un algorithme serveur (qui rend des services) pour gérer des connexions simultanées.

Cet article sert de fil conducteur pour la présentation des notions fondamentales suivantes :



essus (2/5)

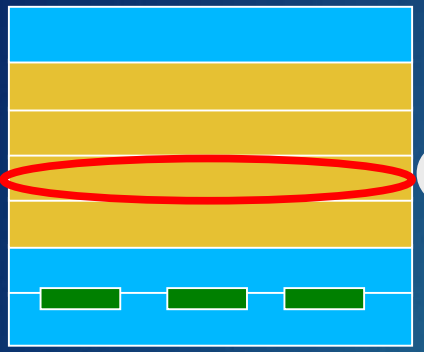
- définition d'un processus (1960), table des processus.
- différentes classes de système d'exploitation, introduction de la notion de « logiciel libre » (projet GNU).
- système multitâche, temps partagé.
- communication : signal, pipe, socket
- « où est le père ? », « où est le fils ? »
- "threads"
- suppression d'un processus (suicide, assassinat)
- contrôle de la communication



essus (3/5)

Les états (simplifiés) d'un processus

- **Waiting** : le processus attend quelque chose pour pouvoir s'exécuter
- **Ready** : le processus a tout pour s'exécuter sauf le processeur.
- **Running** : le processus s'exécute.
- **Zombie** : état très particulier, le processus est mort seul reste son ID dans la table des processus en attente d'être lu …



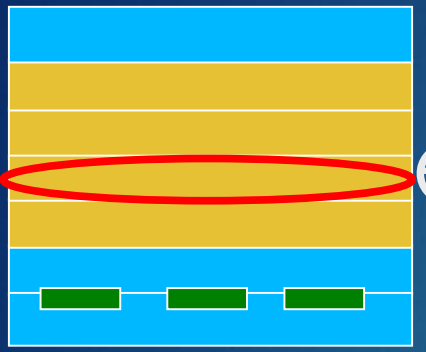
essus (4/5)

Notion de préemption

Définition : la **préemption** est la mise en attente forcée d'un processus.

Un processus spécial s'occupe de faire « tourner » les processus qui sont en *running* : c'est **l'ordonnanceur** ou *scheduler* en anglais. La gestion de la préemption est appelé **l'ordonnancement** ou le *scheduling*.

Les critères de choix appliqué par le S.E. déterminent les performances du système.

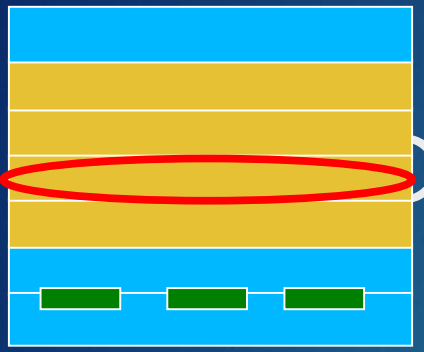


essus (5/5)

Notion de préemption (suite)

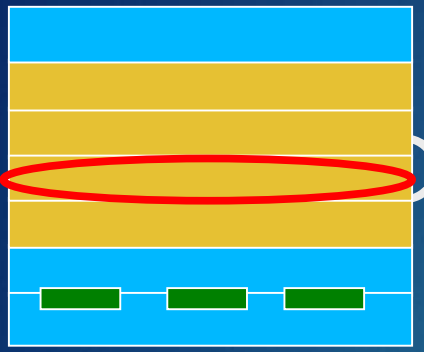
Il y a différent niveau d'attente d'un processus, c'est à dire différents états possibles au sein de l'état *waiting* et de l'état *ready*. En effet, lorsque la mémoire est saturée, le processeur transfère une zone mémoire sur le disque pour libérer de la mémoire. Il choisit la zone mémoire d'un processus qui ne fait rien. Lorsque le processeur devra exécuter un bout du code de ce processus il devra recharger la zone mémoire stockée sur disque en mémoire centrale. Cette opération (dans un sens et dans l'autre) s'appelle le **SWAP** !

D'où deux états supplémentaires : « waiting swappé » et « ready swappé ».



cessus exemple de création – simple 1

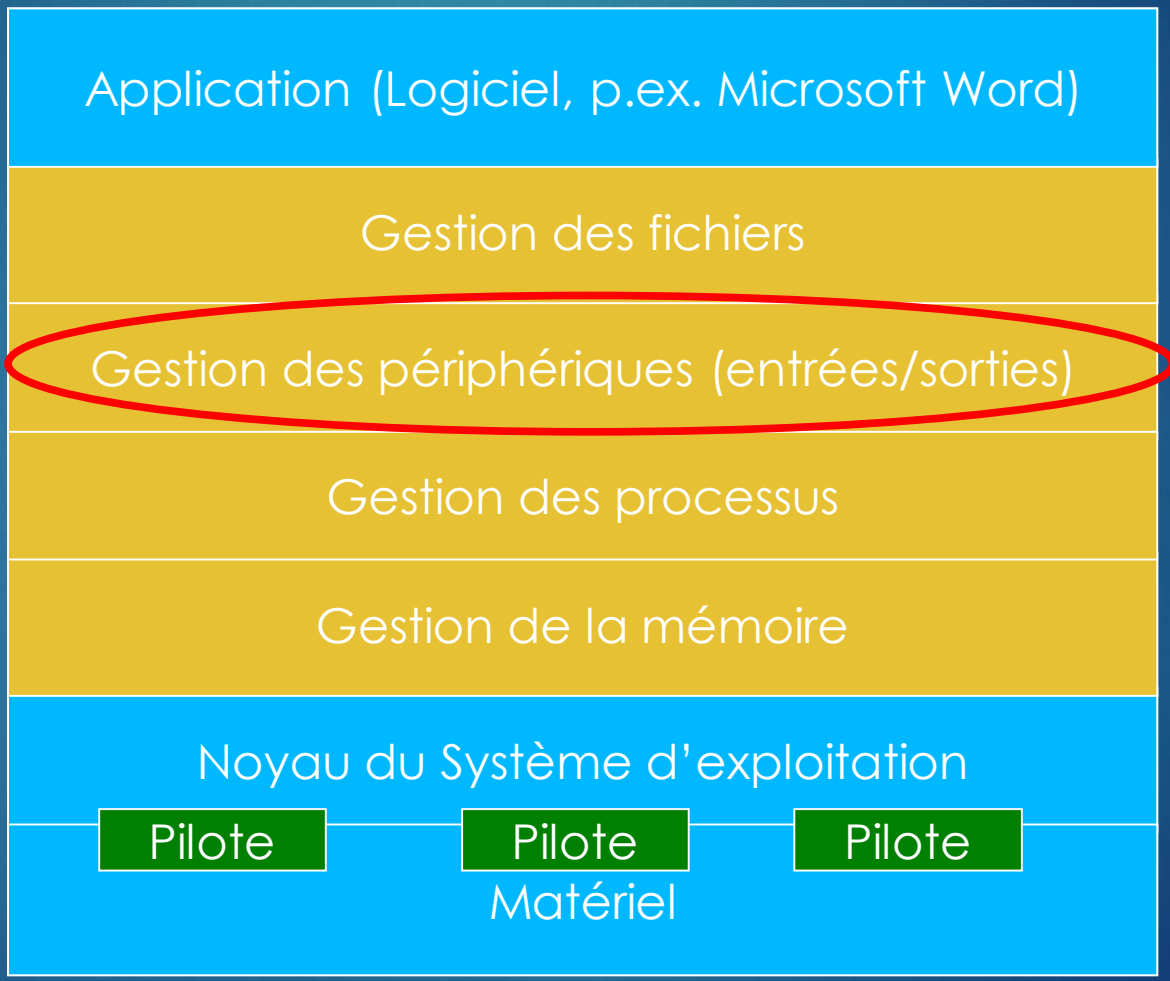
```
#include <stdio.h>
#include <unistd.h>
int main()
{
        printf("Bonjour de %d \n ", getpid());
        fork();
        printf("Fin de %d  \n ", getpid());
}
```

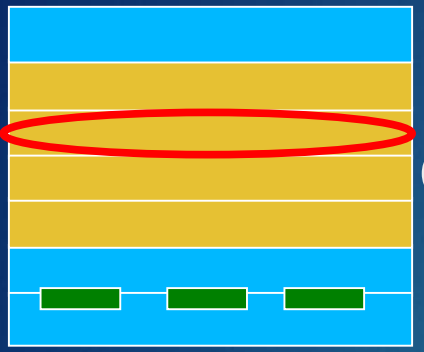


Processus
exemple de création – simple 2

```
#include <stdio.h>
#include <unistd.h>
int main()
{
	printf("Bonjour de %d\n", getpid());
	int pid = fork();
	if (pid == 0)
	{
		printf("Je suis le FILS (pid = %d). %d\n", pid, getpid());
	} else
	{
		printf("Je suis le PERE (pid = %d). %d\n", pid, getpid());
	}
}
```

Modèle en couches



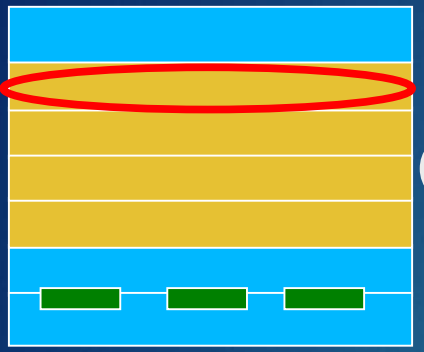


es-sorties

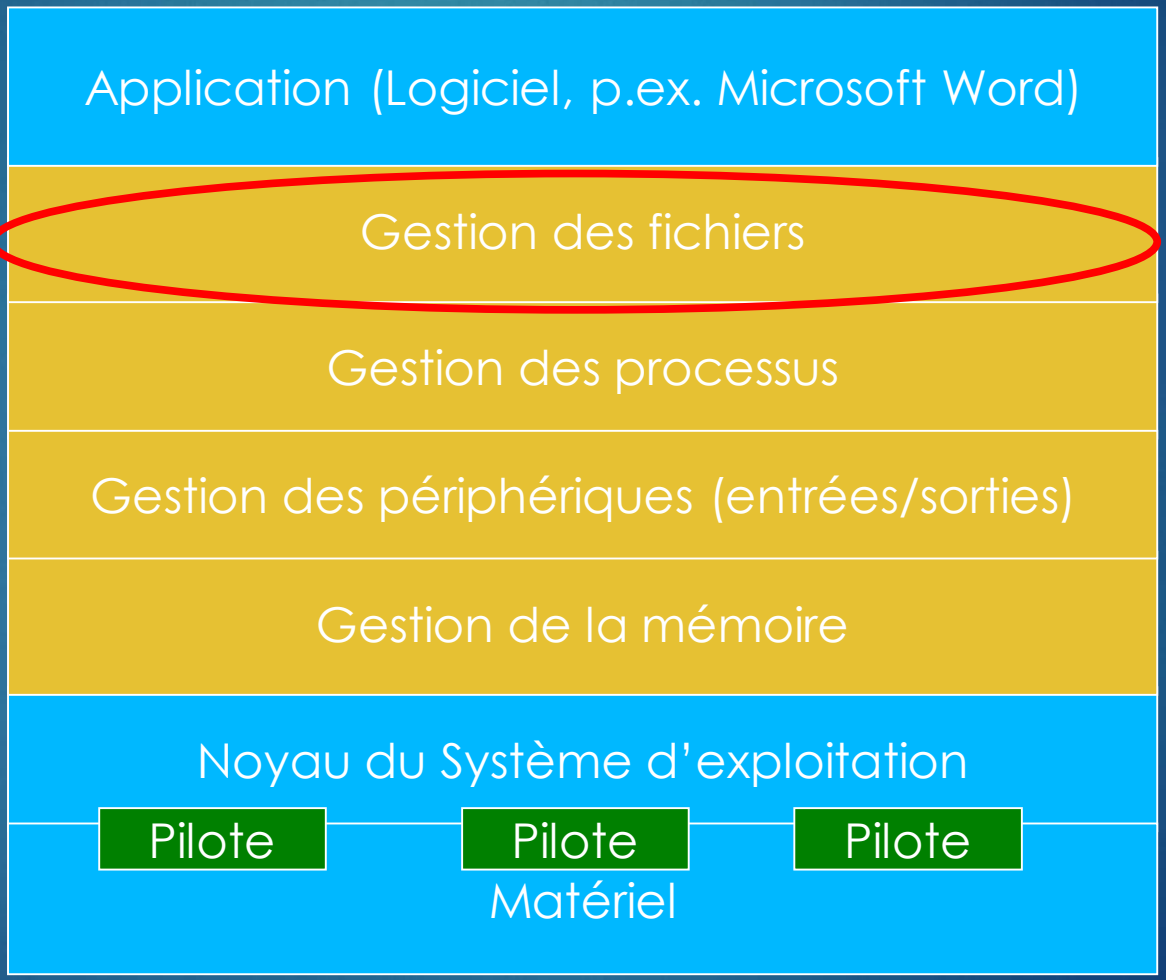
Les **entrées/sorties** correspondent aux mécanismes qu'utilisent les processus pour communiquer avec l'extérieur. Ces entrées-sorties font largement appel aux couches les plus proches du matériel, et dont le système tente de masquer les particularités aux utilisateurs.

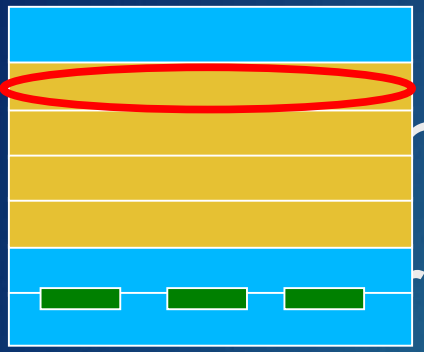
Il y a 3 types d'E/S :

1. Électroniques : mémoires
2. Magnétiques : disques ou disquettes
3. Mécaniques : clavier, imprimantes

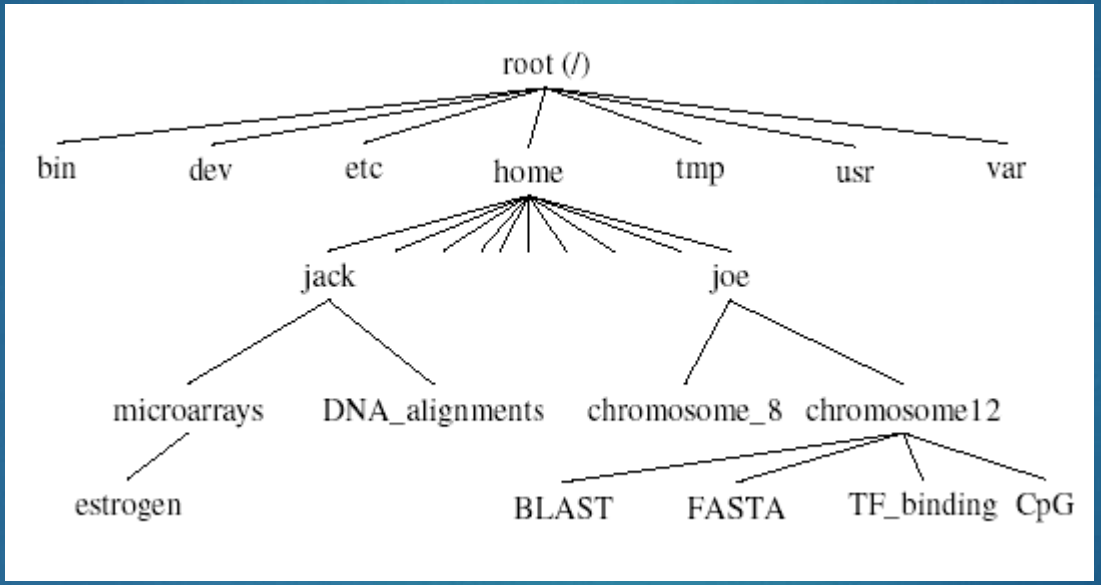


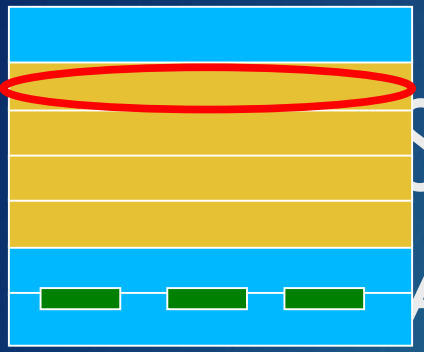
èle en couches



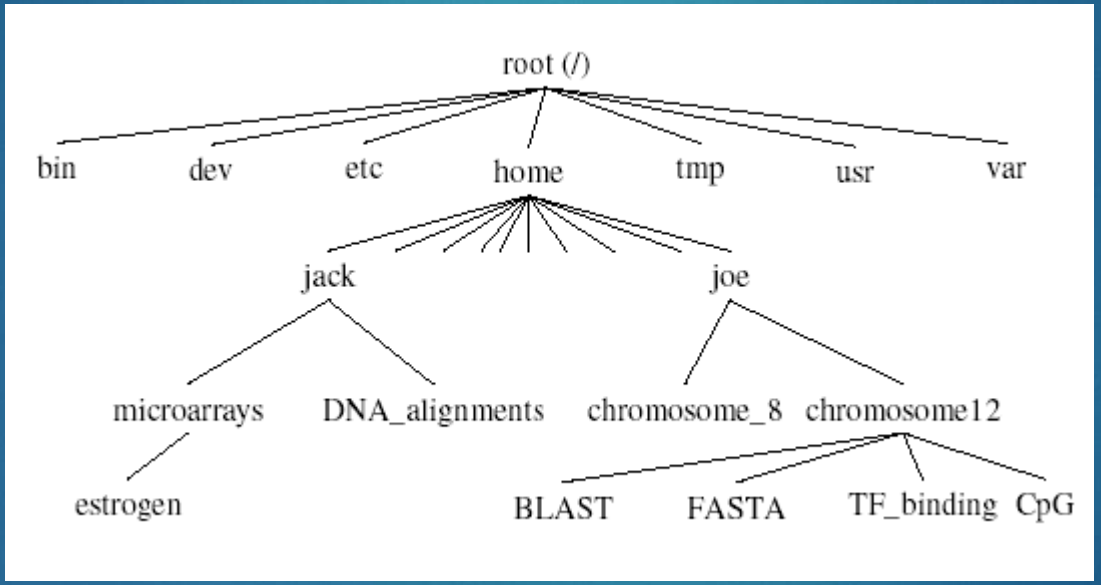


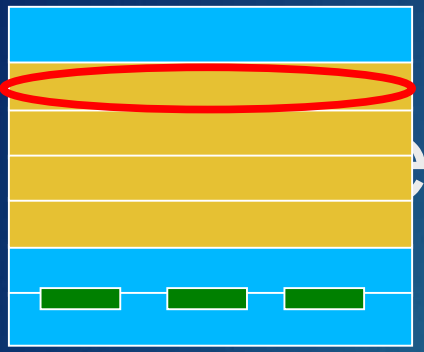
mes de fichiers
escence





Systèmes de fichiers
Arborescence





ème de fichiers

- Arborescence
- Nom des fichiers:
  - **/home/ali/microarrays/estrogen**

```
-rw-r----- 1 karim lipsi 2340 Jun 11 17:45 guethary.jpg
-rwxr-x--- 1 karim lipsi 2340 Jun 11 17:45 Man.exe
```

- Droits:
  - Utilisateur Groupe Tous
  - 'r' lire, 'w' ecrire, 'x' éxécuter
- Types:
  - '-' Fichier regulier, 'd' repertoire, 'x' lien symbolique..

ème de fichiers UNIX
ertoires

- Créer des repertoires

```
mkdir [repertoire]
```

- Changer le repertoire courant

```
cd [repertoire]
cd ..
```

- Connaitre le repertoire courant

```
pwd
```

- Effacer un repertoire

```
rmdir [repertoire]
```

ème de fichiers UNIX
iers

- Créer des fichiers

```
emacs [fichier]
```

- Copier des fichiers

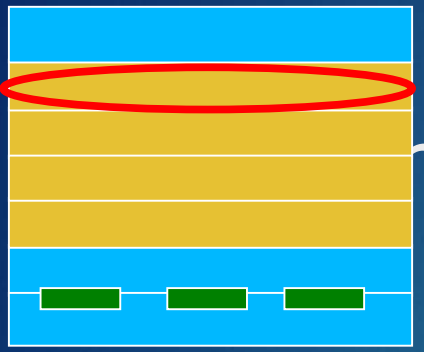
```
cp [source] [destination]
```

- Renommer/Déplacer des fichiers

```
mv [source] [destination]
```

- Supprimer des fichiers

```
rm [fichier]
```



me de fichiers

- Le système de fichier offre à l'utilisateur une vision homogène et structurée des données et des ressources : disques, mémoires, périphériques.

- Le système gère la création des fichiers, leur destruction, leur correspondance avec les dispositifs physiques, ainsi qu'un certain nombre d'autres caractéristiques, telles que la protection .

- Il les organise enfin, en général, en une structure arborescente

conclusion (1/2)

Les systèmes d'exploitation modernes intègrent par ailleurs d'autres caractéristiques ….

- l'interconnexion des différentes machines et des différents systèmes par des réseaux locaux ou étendus d'où des architectures informatiques fondés sur des clients et des serveurs (cf I2-SI)
- Multi-fenêtrage

conclusion (2/2)

Le système d'exploitation correspond à l'interface entre les applications et le matériel !

- Le programmeur d'applications n'aborde que rarement – sinon jamais – son code interne. Il l'utilise par l'intermédiaire d'« appels système », accessibles à partir d'un langage de programmation (lang. C). Ces appels permettent d'effectuer la plupart des opérations sur les entités du système d'exploitation (cf. API (*Application Programming Interface*)).

- Un utilisateur peut lui aussi – dans une certaine mesure – manipuler un système d'exploitation …. interprète de commandes (*shell)*

Références

1. **Computer Architecture, a first course, Van Nostrand Reinhold, ISBN 2 225 80929 1**
2. **Ecrire des applications réseau sous Linux, thème Système et réseaux de Linux Magazine, octobre 99, par Alain Basty.**
3. **man d'Unix,** *notons ici que lorsque l'on doit programmer les systèmes d'exploitation le man d'Unix est la meilleure référence pour le programmeur. Il n'y a rien d'équivalent sur papier*.
4. **J.M. Rifflet, La programmation sous Unix, 3e éd., McGraw-Hill, 1993** est cependant une bonne référence et un ouvrage assez complet
5. **Charles Petzold, Programming Windows 95, Microsoft Press, 1996** est considéré comme une bonne référence pour apprendre la programmation Windows. Elle a été la première du genre. Actuellement, il y a des dizaines d'ouvrages équivalents.