Cours 1 : Introduction aux systèmes d'exploitation

Pourquoi de cours ?

- En tant que développeur, vous devez comprendre le fonctionnement du système exploitation
- Les administrateurs systèmes sont en demande et il existe de nombreux problèmes potentiels associés au système.
- Si vous êtes assez bon pour écrire du code pour un OS**, vous pouvez écrire du code sur presque tout le reste.
- C'est un métier: Ex. Linux Systems Administrator (**69,71 €**)

**https://github.com/github/linux

Objectifs du cours

En savoir plus sur les systèmes d'exploitation et dela programmation système.

Les étudiants deviennent compétents dans:

→ Comprendre le fonctionnement du SE (OS)

→ Comprendre les mécanismes utilisés dans le noyau de système d'exploitation

→ Test des performances des système + Optimisation du système

→ Utiliser les outils de debug pour debugger le noyau

→ Comprendre les différents mécanismes de synchronisation et d'exclusion mutuelle du noyau

À propos de cours

- Contenu de ce cours?
  - Introduction (Historique des Systèmes d'exploitation)
  - Gestion des systèmes de fichiers
  - Gestion de périphériques
  - Gestion des processus
  - Communication et synchronisation interprocessus
  - Gestion de la mémoire
  - Mémoire virtuelle
  - L'ordonnancement des processus
- Déroulement du module
  - 1 cours hebdomadaire (1h 30) sur les principaux concepts.
  - 1 TD hebdomadaire (1h 30).
  - 1 TP hebdomadaire (1h 30).
- Evaluation
  - Contrôle (s) + TP + Examen

À propos de cours

NB:

- Lien de cours: https://github.com/OUARED-A/Operating-System-Courses
- Etudiant au moment de cours encore besoin de *prendre des notes.*
- Noter des questions

…

Plan du cours

Structure des systèmes Informatiques

Système d'Exploitation : Principe

Appels système

Évolution des SE

Modèle en couches de SE

Conclusion

Section 1 : Structure des systèmes Informatiques

Architecture de Von Neumann

- 1945 : modèle de Von Neumann

  Problème: Le programme peut prendre des décisions selon des résultats intermédiaires

  Solution : stocker le programme dans la mémoire de l'ordinateur

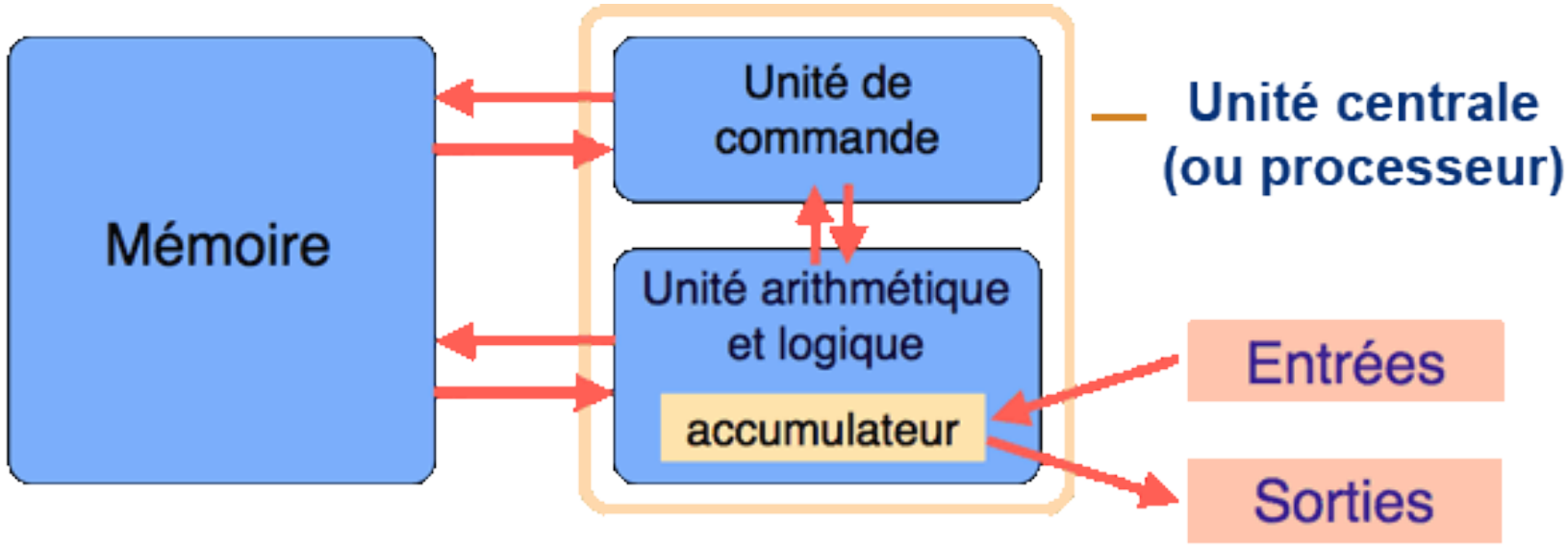
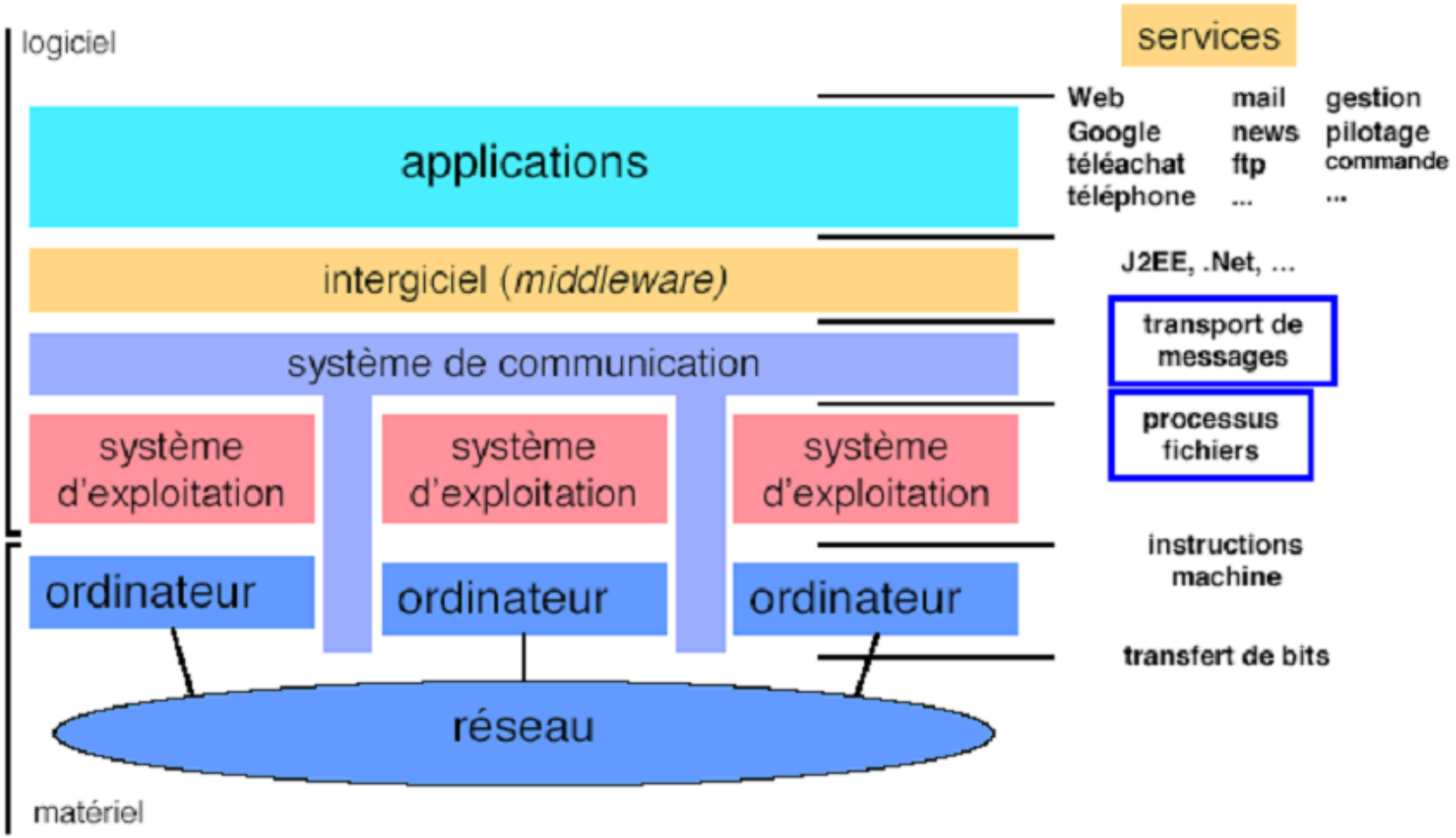


Schéma d'un système informatique moderne



Hardware & Software

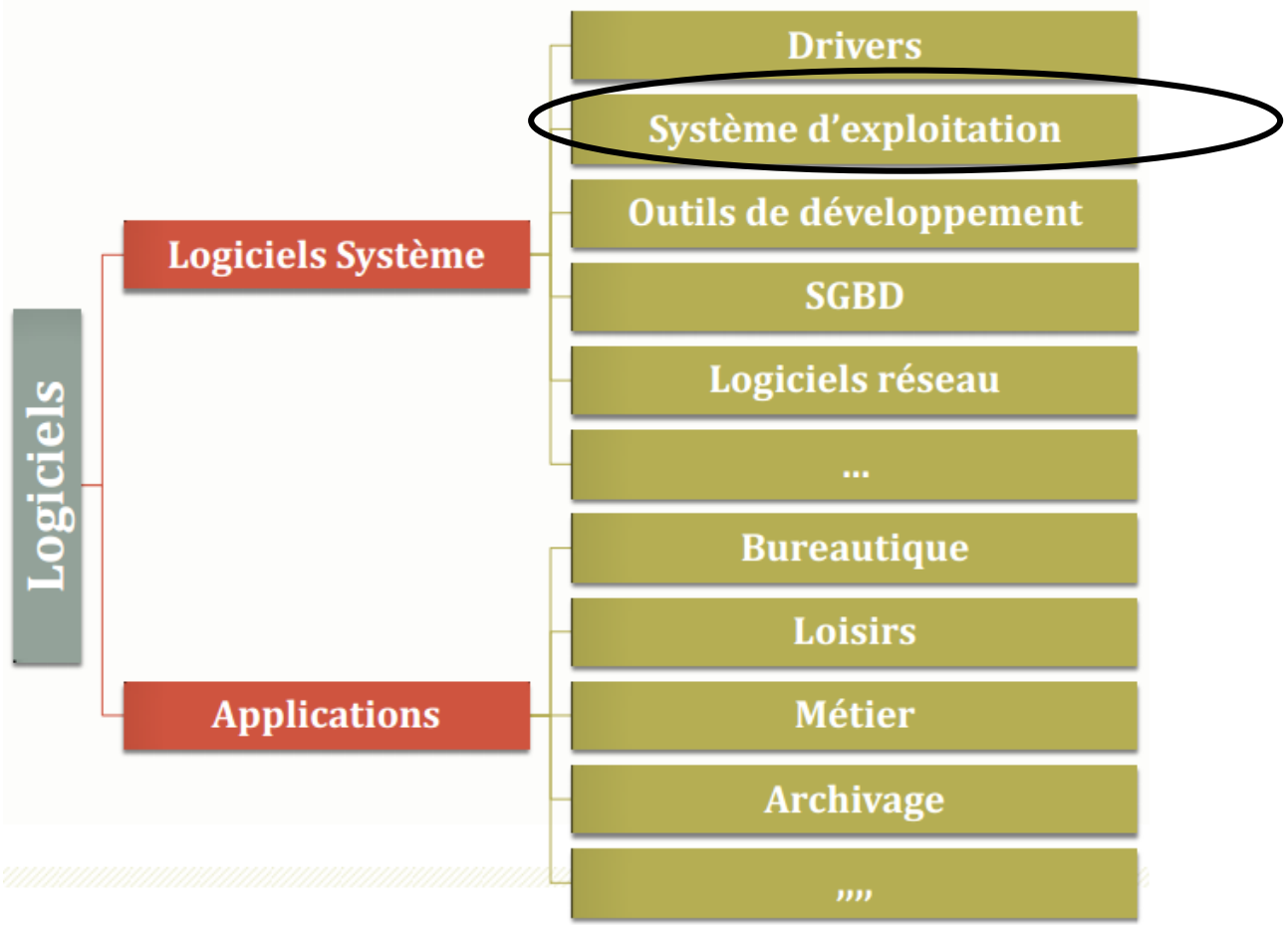
L'ensemble des composants physiques (matériel = hardware) est commandé par un programme (logiciel = software).

Le « software » et le « hardware » sont complémentaires:

- **Le « hardware » a besoin du « software » pour être piloté**
- **Le « software » a besoin du hardware pour être exécuté**

Où se trouve le logiciel (Software)?

Classification de Logiciels



Section 2 : Système d'exploitation: Principe

C'est quoi les res ources ?

angl. « Operating System (OS) »
Qu'est-ce que c'est? « Programme assurant la gestion de l'ordinateur et de ses périphériques » [www.dicofr.com]

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

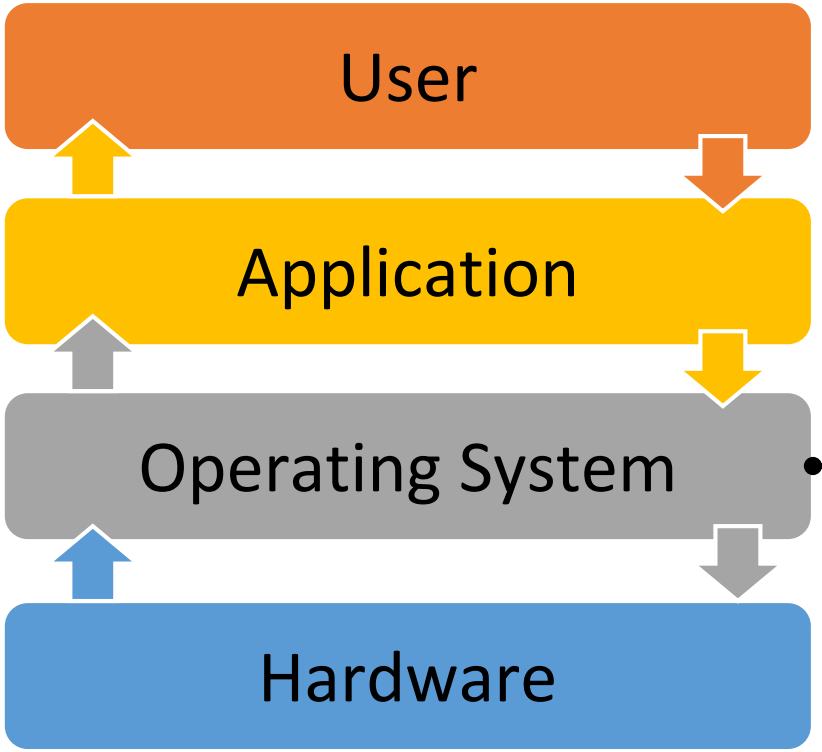
Cacher la complexité des machines pour l'utilisateur afin d'utiliser la machine sans savoir ce qui est derrière
Abstraction du terme « Machine » selon Coy

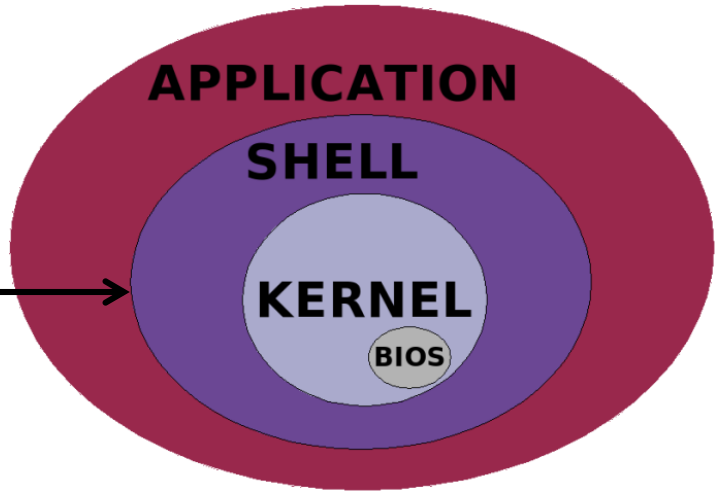
machine réelle = Unité centrale + périphériques (CPU, Mémoire, I/O)

machine abstraite = machine réelle + système d'exploitation

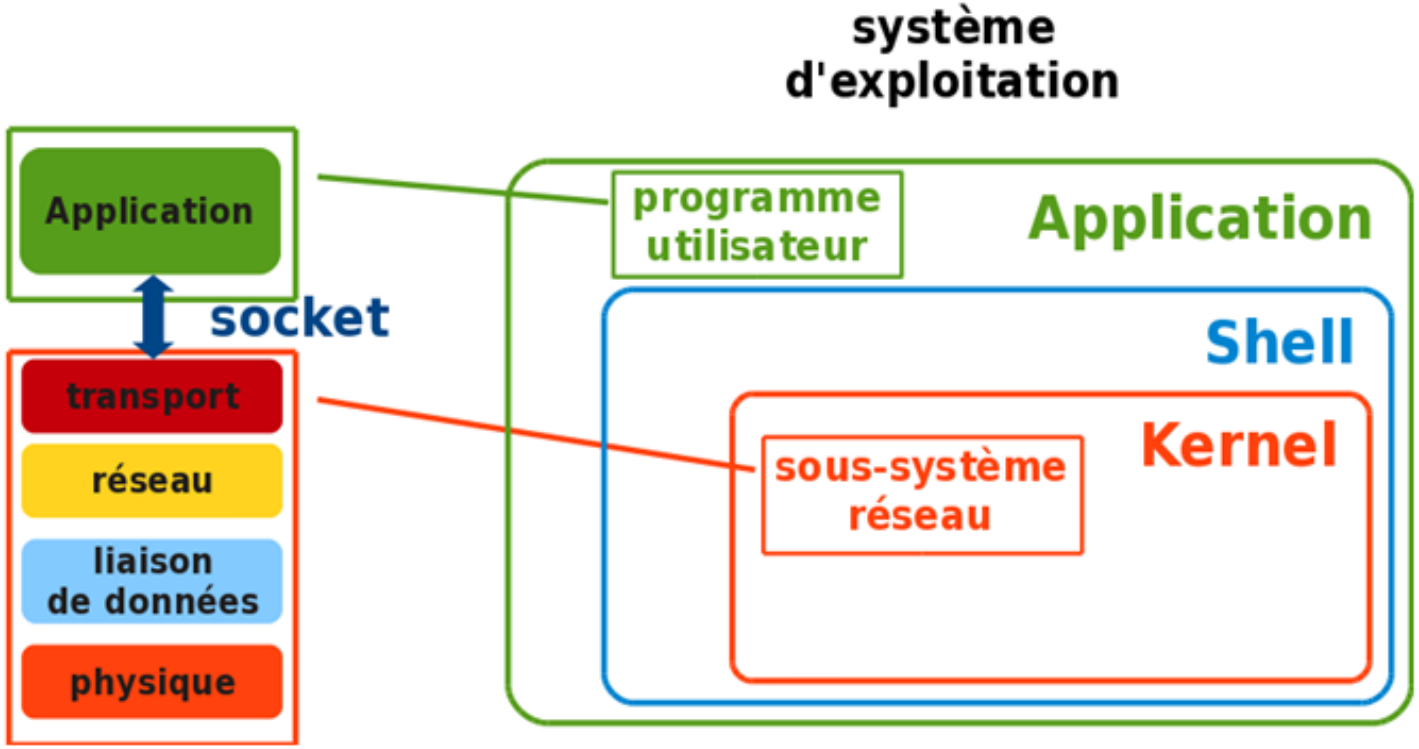
machine utilisable = machine abstraite + application

Abstraction



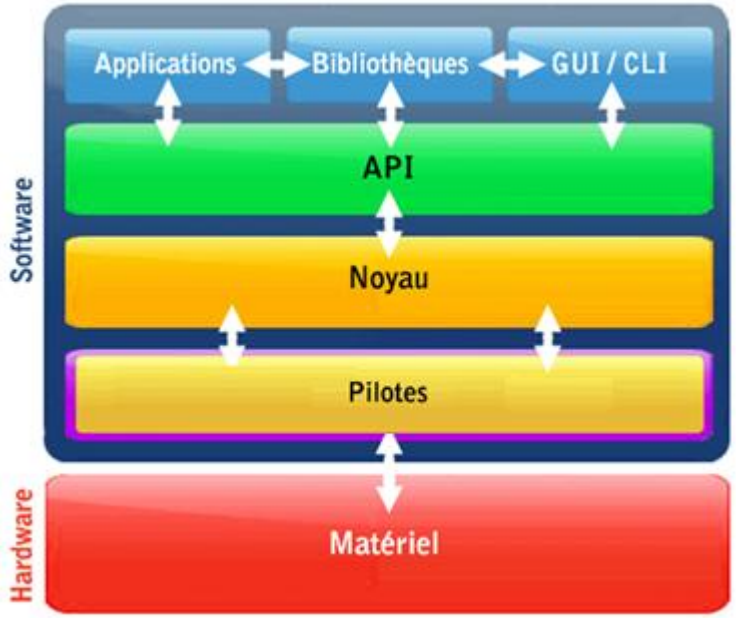


Abstraction
systèmes d'exploitation réseaux

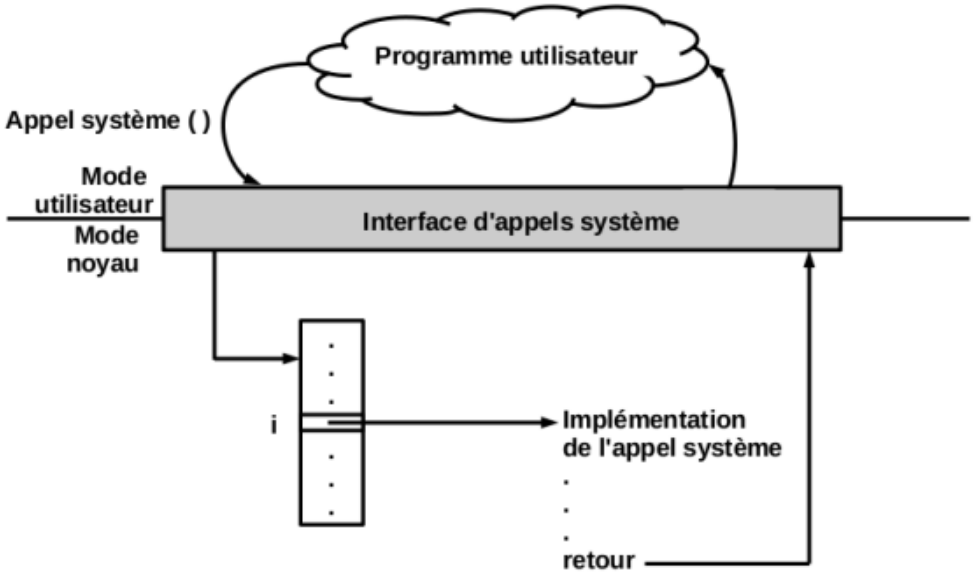


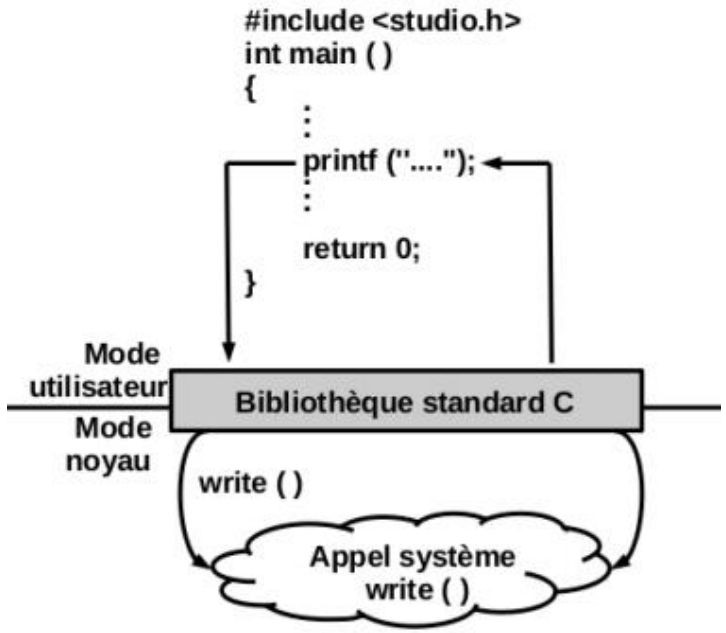
Appels Système

- Interfaces aux services offerts par le SE
- Ecrits en C/C++
- Généralement accessible a travers des bibliothèques de haut-niveau (API)
- Les API les plus utilisées :
  - Win32 API
  - Java API
  - …



Implémentation d'appels système





Modes d'éxécutions

- Mode utilisateur
- Mode noyau

Passage de paramètres aux appels système

- Registre
- Block de mémoire
- Pile

Implémentation d'appels système

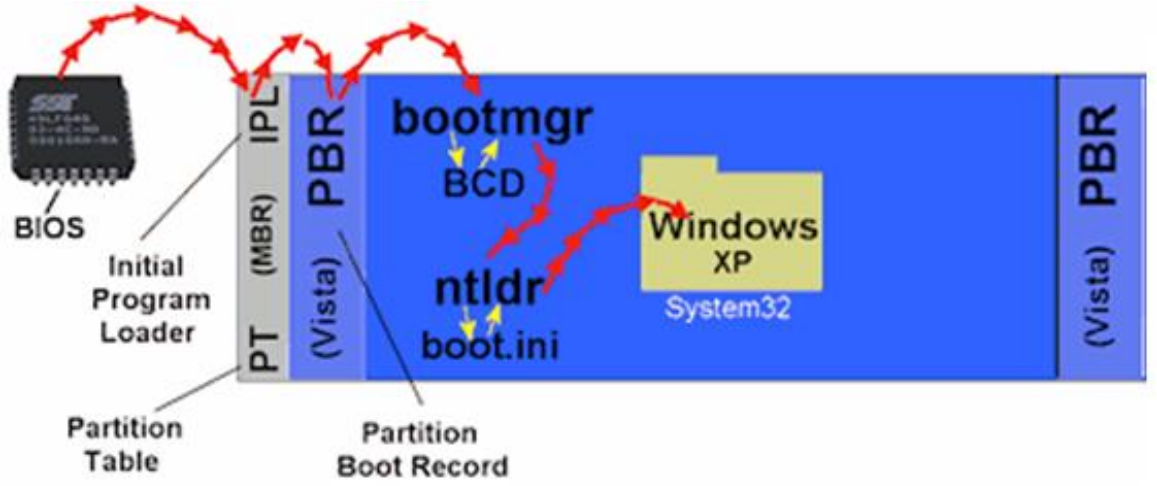
Q1. Expliquez le mécanismes des appels systèmes à travers un exemple. Utilisez un petit schéma illustratif.

Q2. Quel est l'intérêt des appels systèmes, pourquoi ne pas utiliser des simples appels aux fonctions.

Q3. Comment peut-on être sûr qu'aucun programme ne peut contourner le mécanismes des appels systèmes.

Amorçage d'une machine (cas de MS Windows)

BIOS
CMOS ROM
CMOS Setup
Power-On Self-Test (POST)
MBR: Master record Boot/ PBR
Loader (NTLDR): charger NTDETECT.COM et boot.ini
NTLDR: lit boot.ini et exécute le système d'exploitation
NTLDR: identifie l'environnement physique de l'ordinateur



Exercice 01

Ecrire un algorithme ou bien élaborer un organigramme de l'amorçage d'une machine

Section 2 : Système d'exploitation: Principe

Section 3 : Evolution des systèmes d'exploitation

- Un SE s'évoluera au fil du temps pour des raisons:
  - Mise à niveau du matériel
  - Nouveau type de matériel
  - Nouveau service

Historique (avant les Systèmes d'Exploitations)

- 1945 - 55 : tubes et interrupteurs
  - Pas de Système d'Exploitation (utilisant des relais mécaniques)
- 1955 - 65 : transistors, cartes perforées
  - Traitement par lots
- 1965 - 80 : circuits intégrés, disques
  - Multiprogrammation, temps-partagé, entrées/sorties
  - Unix, version BSD, AT&T, interface POSIX
- 1980 : ordinateurs personnels (PC)
  - Interface graphique (concept crée vers 1960, Stanford)
  - Réseaux et systèmes distribués

  --> Système d'Exploitation nécessaire

Systèmes d'exploitations

Windows

Windows 3.11 – pas de multitâche, pas de multi-utilisateurs

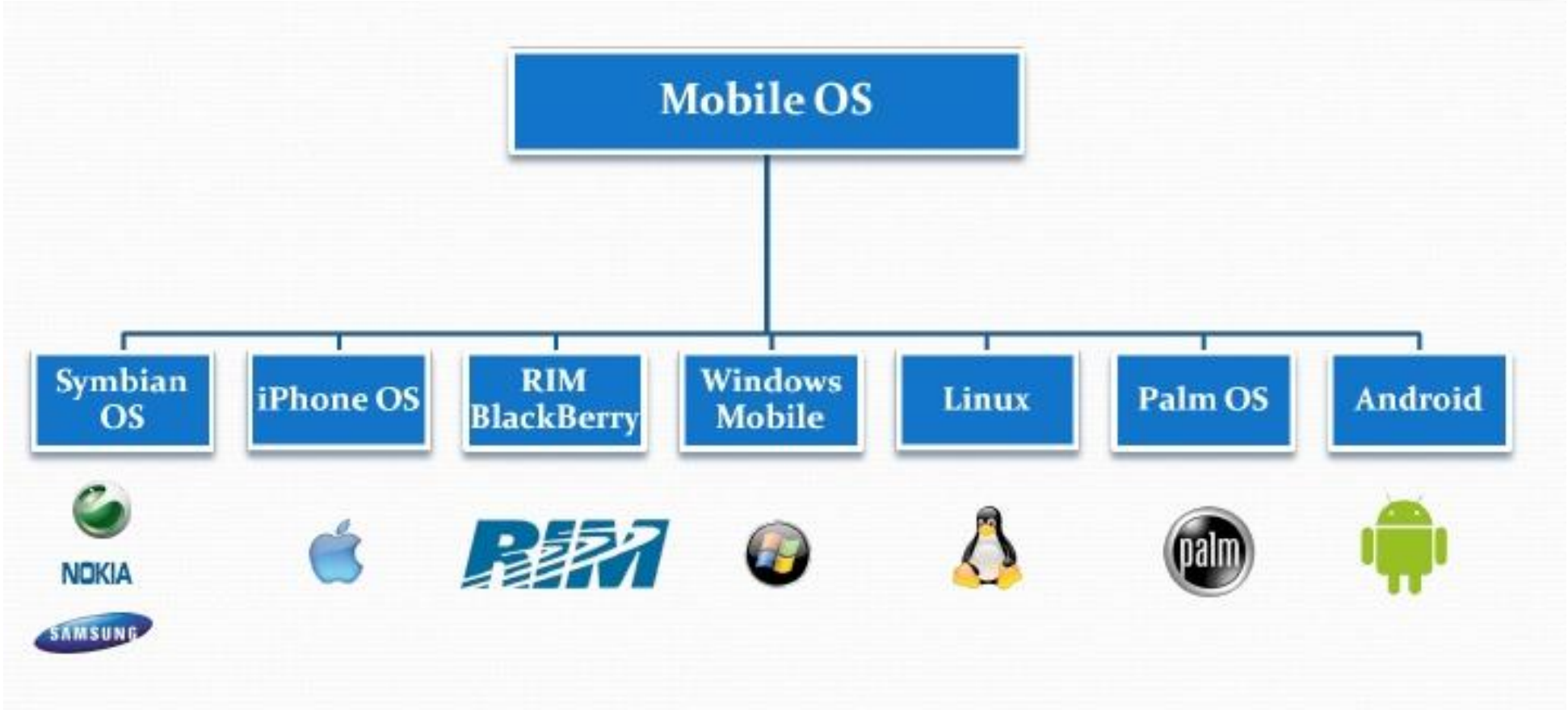
Windows 95 – multi-tâche premier système 32 bit

Windows 98 – Internet integré dans le GUI Plug & Play parallèlement
Windows NT – Système d'Exploitation réseaux multi-utilisateur
Windows 2000, et après Windows XP/7/8…

Linux (depuis 1992), OpenSource – finlandais Linus Thorwald
– Licence GPL (General Public Licence)
– OpenSource
– Multi-tâche et Multi-utilisateurs
– Distributions Red Hat, Fedora, Debian

Lunix vs Unix : recursive acronym for "*Linux* Is *Not UniX*"

Systèmes d'exploitations mobile



Classification des SE

Traitement par lots

Systèmes Multi-tâche

Systèmes Multi-utilisateurs

Systèmes Multi-processeurs

Systèmes temps réel

Systèmes distribués (répartie)

Définitions

Processus Déf.:

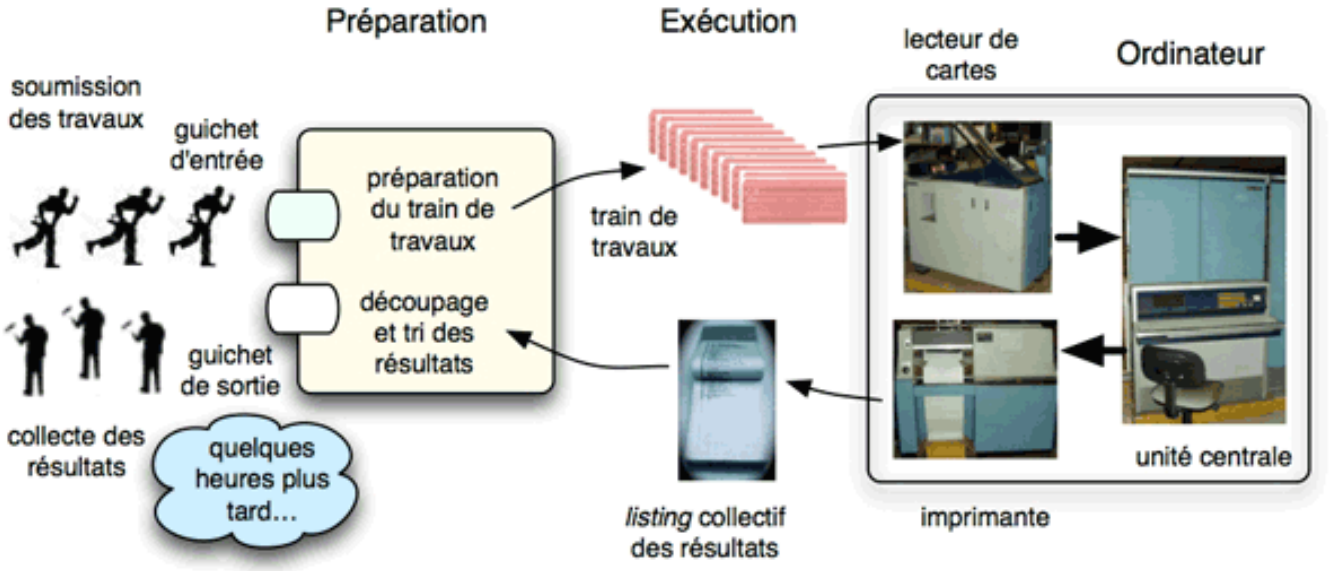
Un processus est un programme lors de l'éxécution

(aspect dynamique d'un programme)

Traitement par lots (Batch processing)

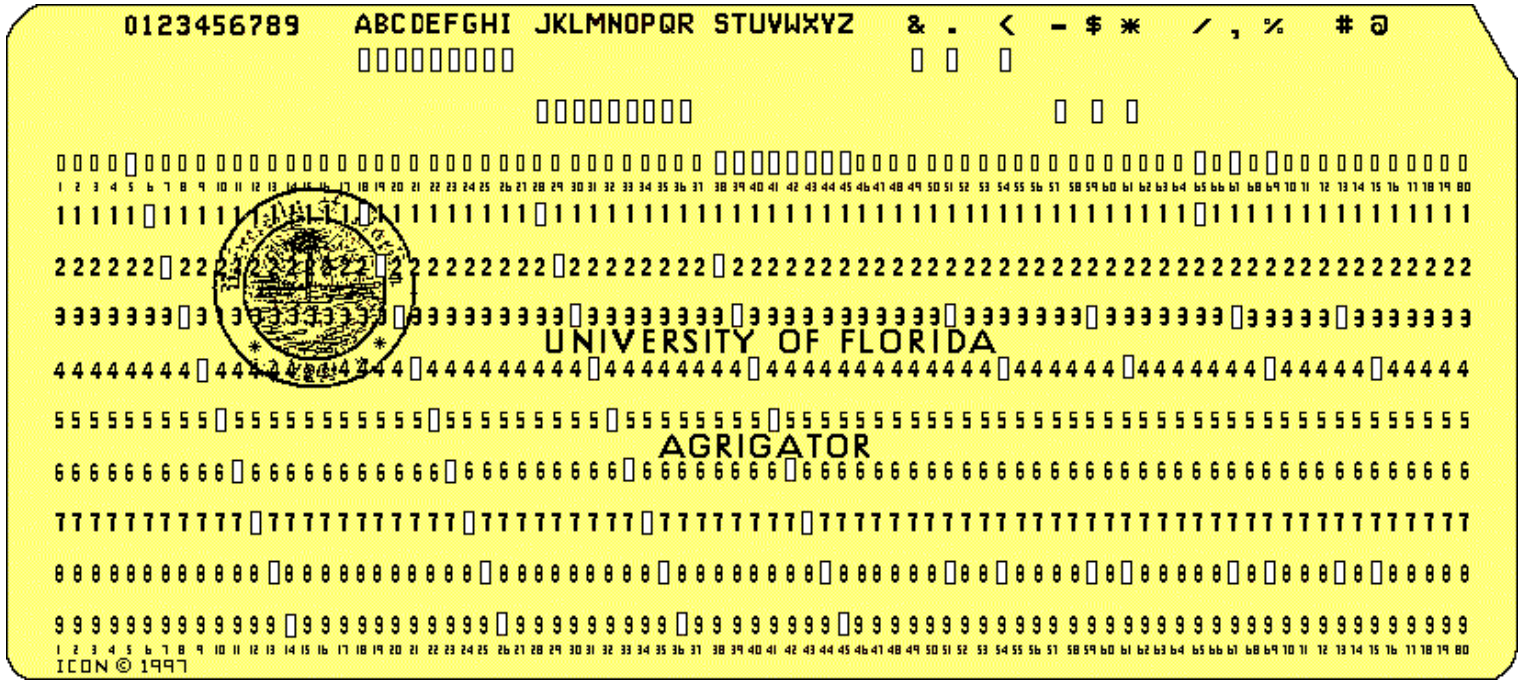
Batch Processing System ou **système de traitements par lots**.

- Premier véritable système d'exploitation.
- Ce Système d'Exploitation est un programme résident en mémoire centrale
- Les travaux successifs sont regroupés en un seul paquet de cartes inséré dans le lecteur de cartes par l'opérateur



Mode de traitement par lots

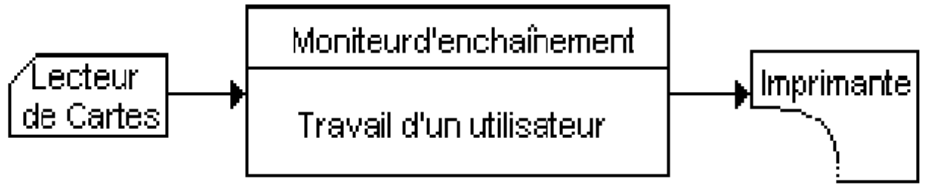
Les cartes perforées…



La carte perforée apparait comme un moyen pratique pour stocker et transmettre des informations.
Une ligne de données ou de programme était codée dans des trous qui pouvaient être lus par la machine

Traitement par lots (Batch processing)

- Le système:
  1. lit un travail,
  2. le place en mémoire centrale, exécute un branchement vers la première instruction exécutable.
  3. A la fin de l'exécution, le travail doit appeler le moniteur qui lit le programme suivant.



- La communication entre l'utilisateur et le moniteur d'enchaînement de travaux est assurée au moyen d'un **langage de commande** qui permet à l'utilisateur de définir le début et la fin d'un travail et préciser les tâches à exécuter.

Traitement par lots (Batch processing)

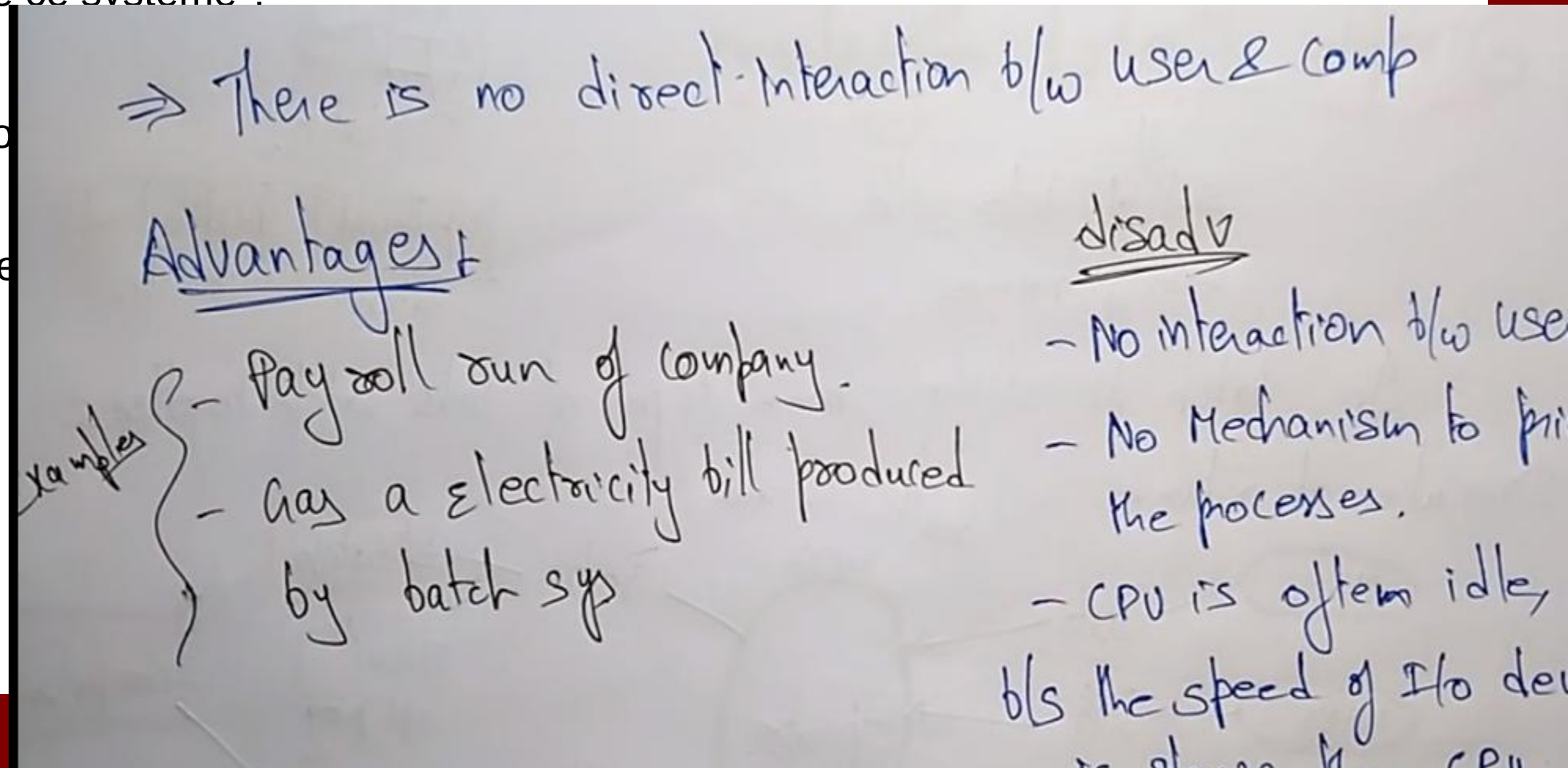
Un utilisateurs donne plusieurs commandes (« Jobs ») dans une queue d'éxécution de programmes

- Entièrement séquentielle
- p.ex. pour faire plusieurs calculs pendant la nuit

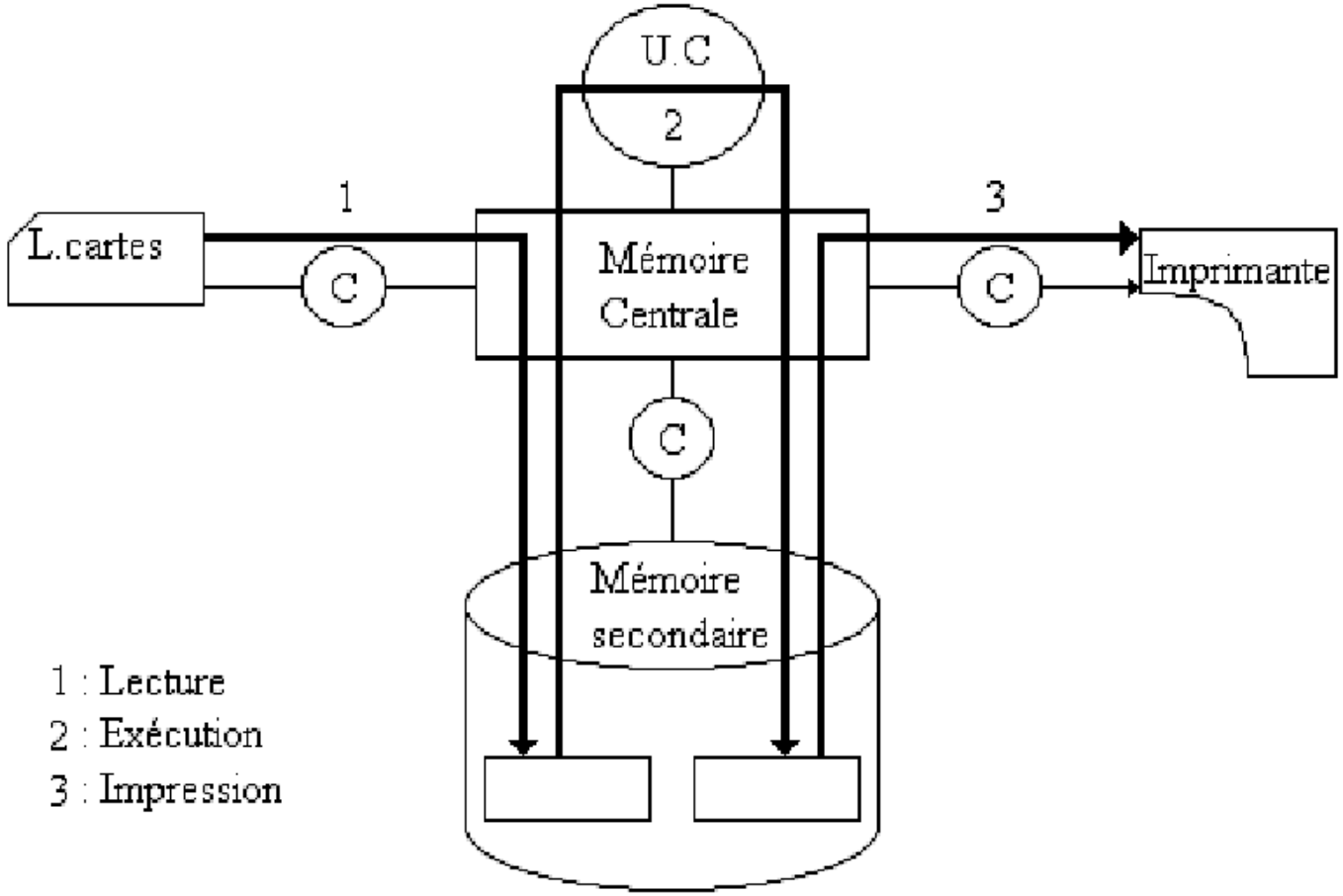
- p.ex. **autoexec.bat**

Traitement par lots (Batch processing)

- Inconvénients de ce système ?
- Comment amélio
- Comment adapte l'UC ?



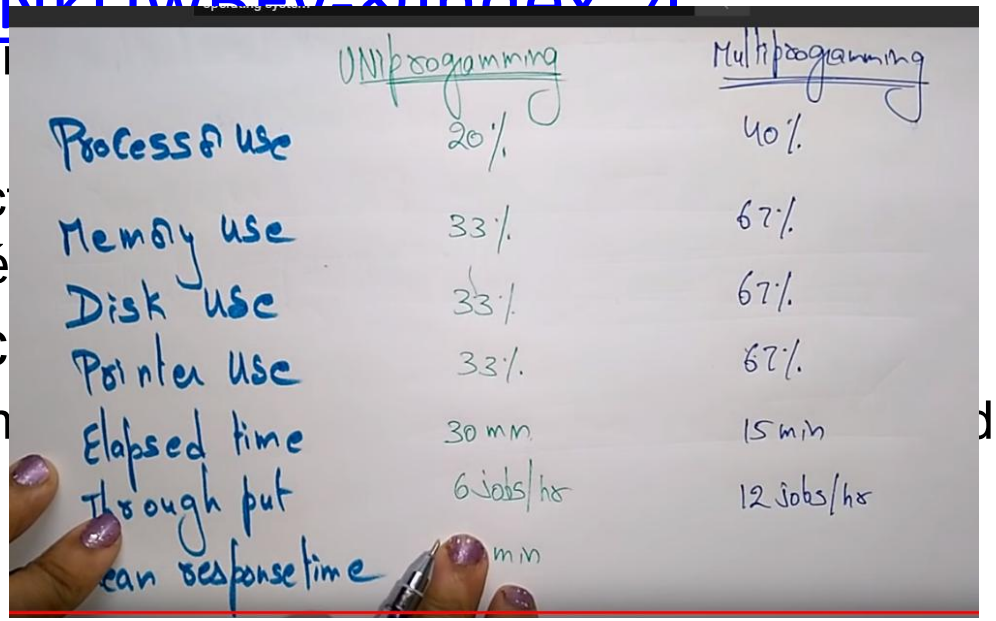
Traitement par lots (E/S tamponnées)



Flux d'information dans un système informatique

Monoprogrammation et multiprogrammation

- Charger en mémoire qu'un seul programme pour l'exécuter ☹ https://www.youtube.com/watch?v=Cv8vu-H6imU&list=PLrjkTql3jnm9U1tSPnPQWQGIGNkUwBEv-&index=4
- Solution : Multi programmation ☺
  - La présence simultanée, en mémoire pi
  - Affectation de processeur
  - Le processeur pourrait changer d'affec
    satisfaire des contraintes de temps de ré
  - la multiprogrammation nécessite c
    - la protection de la mémoire, le systèn des ressources…etc.



Systèmes Multi-tâche

Assurer l'éxécution de **plusieurs programmes (application)** en même **temps**

(c-à-d. plusieurs processus)

Chaque processus a besoin du processeur

- situation concurrente
- solution: « *scheduling* »

Systèmes Multi-processeurs

Système avec plusieurs processeurs

- Parallèle (>= 2CPU)
- Vrai multi-tâche
- Doit assurer qu'il y a l'éxecution d'autant de processus que processeurs en même temps

- Contrairement: système avec un seul processeur
  - Quasi-parallèle
  - Arrêter et reprendre les différentes processus
    - Gestion avec le « *scheduler* » (ordonnancement des processus)

Systèmes Multi-utilisateurs (temps partagé)

Permettre a **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

Systèmes Multi-utilisateurs (temps partagé)

La technique du temps partagé consiste à offrir à chaque utilisateur l'équivalent **d'une machine individuelle,** tout en faisant bénéficier des services communs.

- **Comment** ? Accès par des terminaux de manière interactive ;
- **Condition** : Garantir à chaque utilisateur un temps de réponse acceptable (de l'ordre de la seconde)
- **Méthode** : Allouer successivement le processeur par tranches de temps très brèves (ou quantum) aux utilisateurs ;

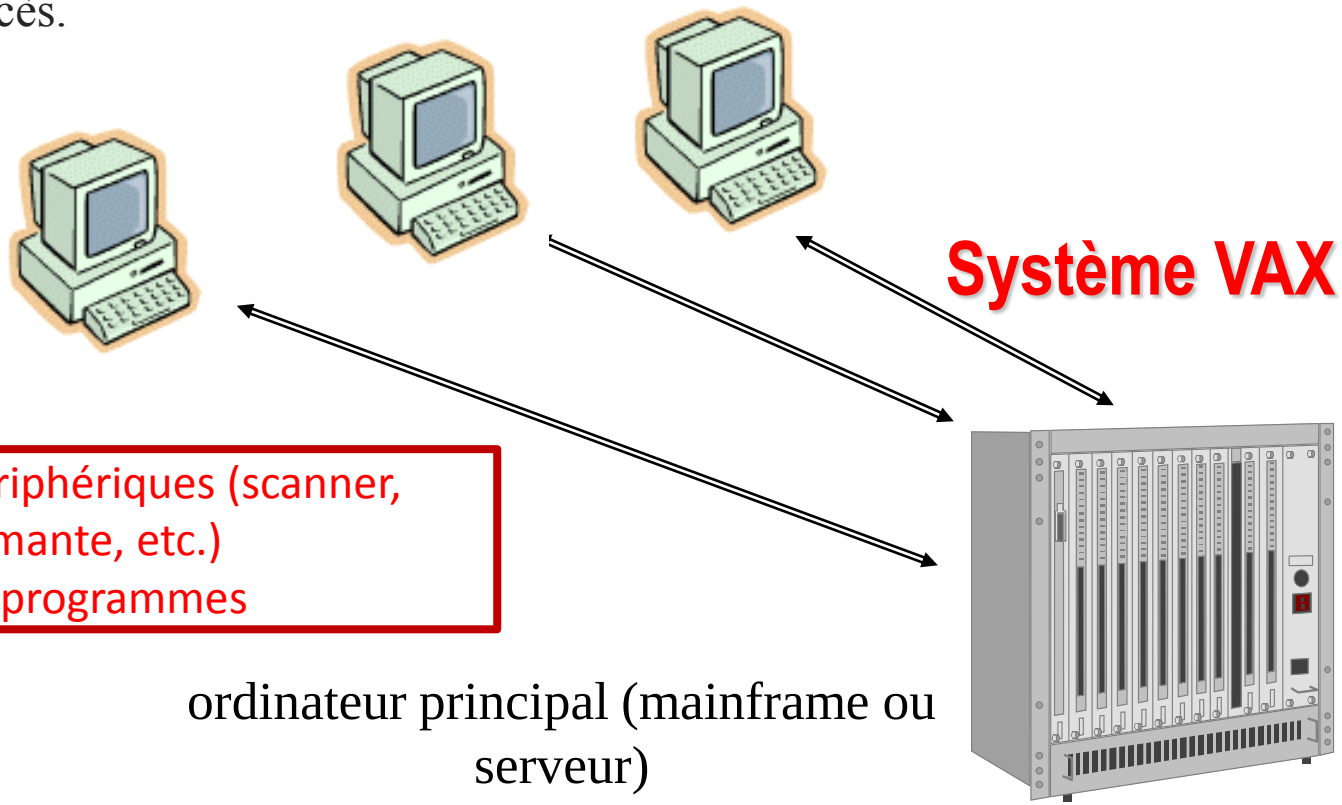
Il est évident que la présence de plusieurs utilisateurs dans le système implique aussi la **multiprogrammation**.

Systèmes Multi-utilisateurs (temps partagé)

UNIX est un système conçu pour laisser plusieurs personnes se servir d'un seul et même ordinateur en même temps

Terminaux reliés à un seul ordinateur

Unix a introduit des mécanismes de contrôle d'accès.



Systèmes Temps réels

Sert pour le pilotage et le contrôle des déroulements externes

(p.ex. centrale électrique, systèmes de pilotage des réacteurs nucléaires, systèmes de défense du territoire).

doit garantir des temps de réactions données pour des signaux extérieur urgents

Les systèmes temps réel souples ont des contraintes temporelles moins strictes et ils ne supportent pas le scheduling d'échéances

Systèmes distribués

Définition. Un système reparti est un ensemble de processeurs ne partageant pas de mémoire ou d'horloge.

- 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

Systèmes distribués

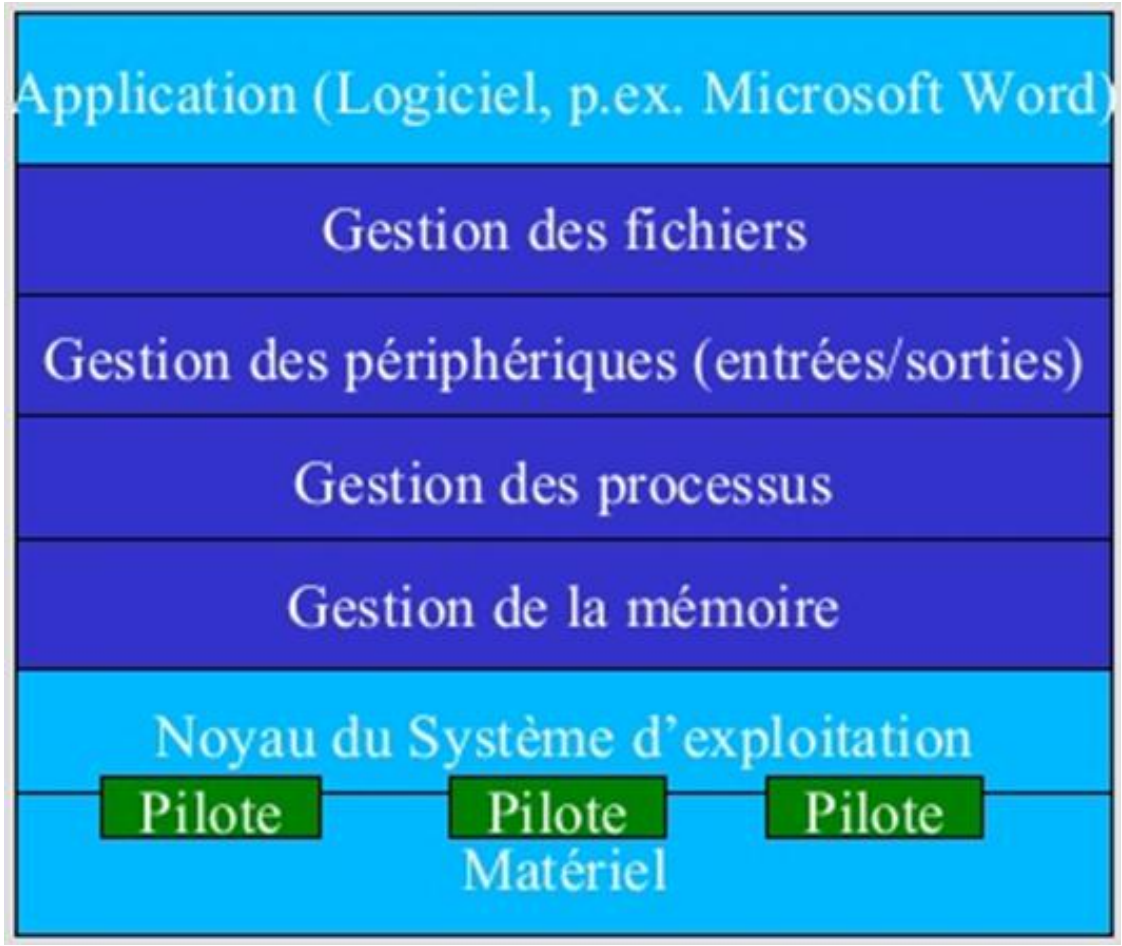
Origines. Les années 80 ont vu le développement de deux technologies :

- L'apparition des microprocesseurs et l'accroissement de leurs performances, qui permet de disposer d'une grande puissance de calcul à des coûts de plus en plus faibles,
- Le développement des techniques de transmission de données (téléinformatique) et l'intégration progressive de la fonction de communication dans les systèmes informatiques

Historique

Section 4 : SE: Modèle en couches

SE: Modèle en couches



SE: Modèle en couches

Section 5 : Conclusion

Conclusion

SE est quasiment partout

SE est Utilisé pour Exploiter la machine

MU=Applications +MV

SE a connu une grande évolution

Atelier – 1

installer Ubuntu sur votre machine

Atelier – 2

Utiliser Windows PowerShell

Bibliographie

Bibliographie

- **[Kaiser, 2006]** : est un cours du CNAM qui évoque les concepts évoqués dans CSC4508/M2, sous la forme d'un document rédigé. Disponible sur Internet.
- **[Tannenbaum, 2001]** : une référence dans le domaine de la conception des systèmes d'exploitation.
- **[Bloch, 2008]** : présente l'histoire des systèmes d'exploitation, leur fonctionnement et les enjeux.
- **[Downey, 2005]** : une référence pour tout ce qui concerne les paradigmes de synchronisation.