IFT-6800, Automne 2016

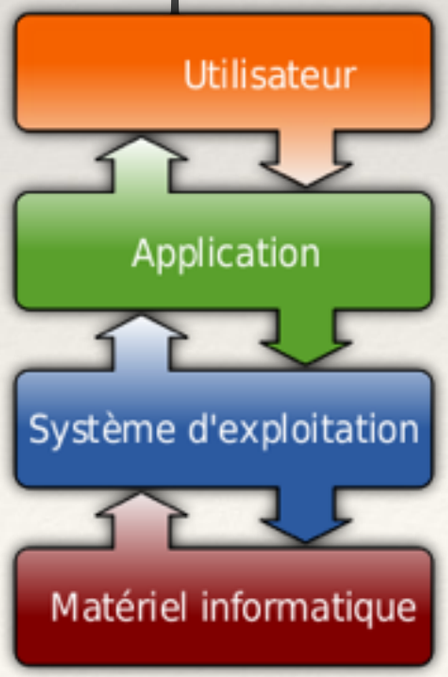
Cours #3–Les systèmes d'exploitation

Louis Salvail
André-Aisenstadt, #3369
salvail@iro.umontreal.ca

Système d'exploitation: C'est quoi?

- Un système d'exploitation (SE, OS en anglais) est un ensemble de programmes responsables des liaisons entre les ressources matérielles d'un ordinateur et les applications de l'utilisateur (traitement de texte, base de données, jeu vidéo, …)
- Il fournit aux applications des points d'accès génériques pour les périphériques.
- Il garantit la sécurité des données.

dispositifs connectés à un système informatique pour lui ajouter des fonctionnalités. Il y des périphériques d'entrée et de sortie. Il y a aussi des périphériques qui font les deux (ex. clé USB).



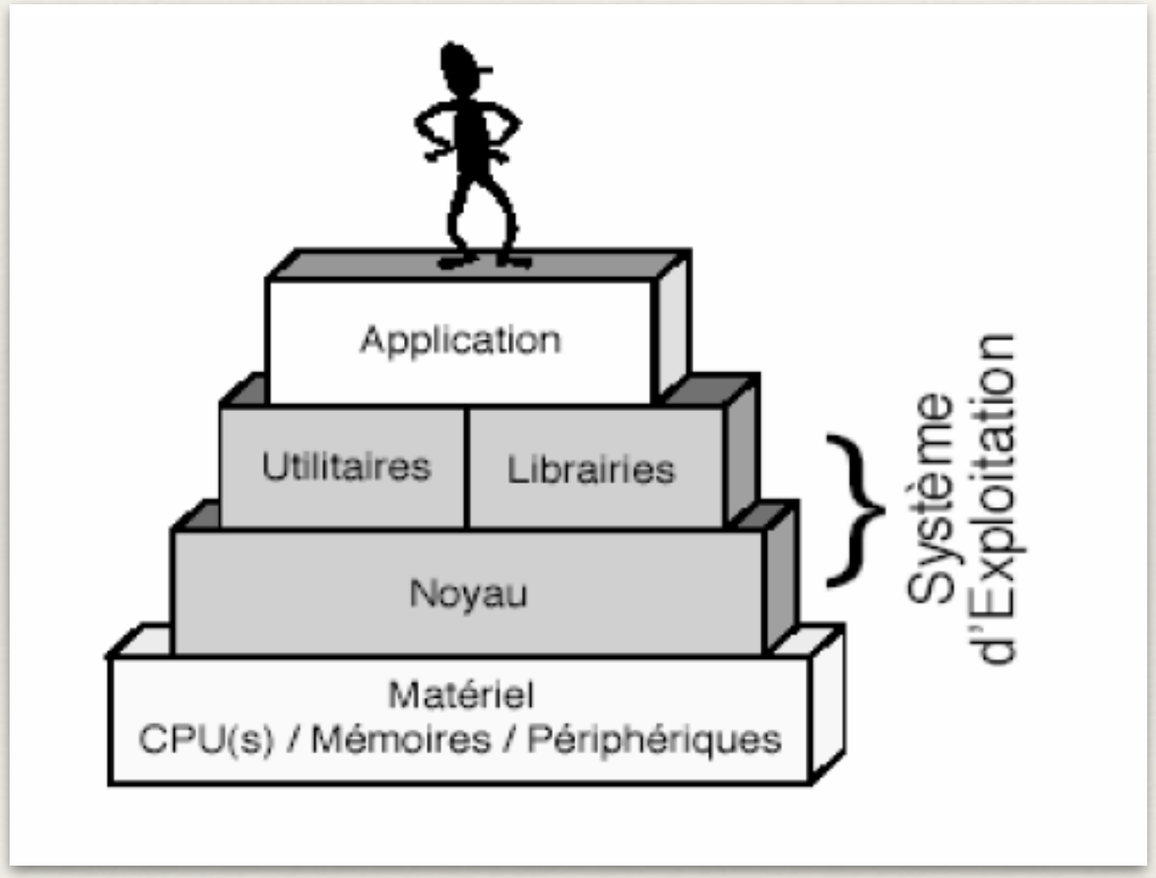
Son rôle

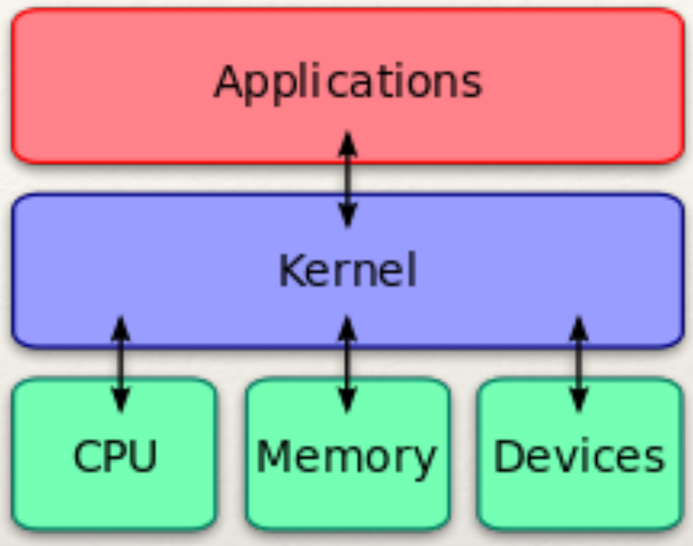
- Il s'agit du maître d'orchestre pour utiliser toutes les ressources de la machine:
  - Gestion de la mémoire centrale.
  - Gestion des processeurs: comment séparer les calculs demandés par plusieurs programmes qui s'exécutent simultanément.
  - Gestion des périphériques d'entrée/sortie.
  - Gestion des fichier sur les mémoires de masse.
  - Gestion des ressources: attribution des ressources nécessaires à un programme.
  - L'exécution du programme sans empiétement…
  - Gestion des utilisateurs: Éviter qu'ils interfèrent entres-eux.

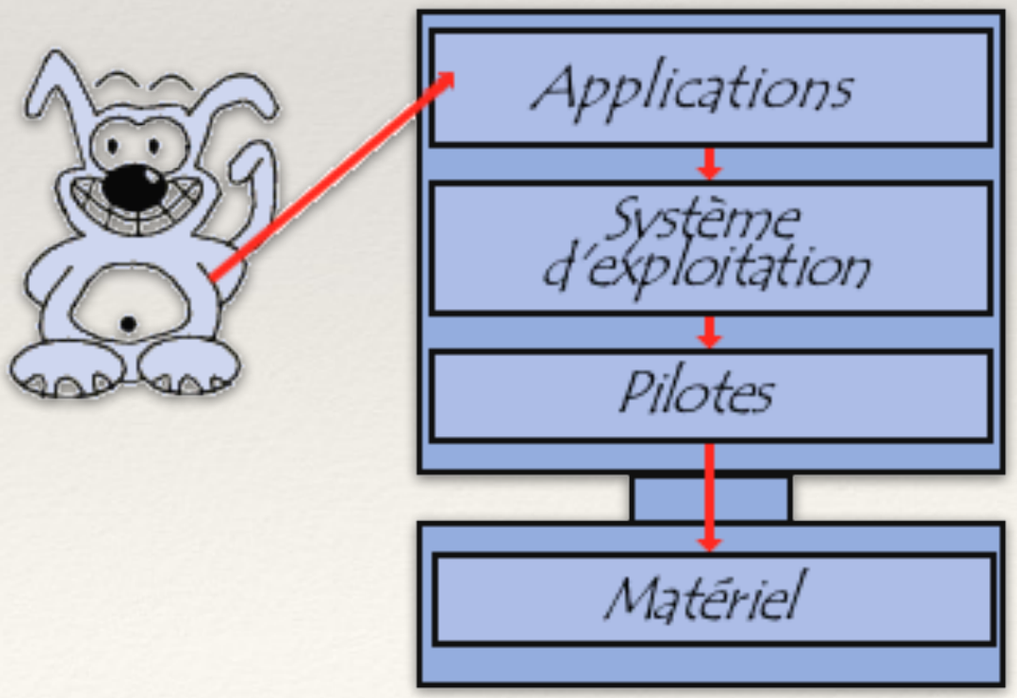
Composition

- Un système d'exploitation (SE) est habituellement composé de:
  - **Un noyau:** Contient la base du SE. C'est un programme qui code les fonctions de base du système. L'administration de la mémoire, des périphériques, les appels systèmes (une application demande un service au SE que seul le SE peut accomplir). Il est habituellement rangé à un endroit en RAM qui ne peut pas être modifié (protégé).
  - **Des bibliothèques dynamiques:** Des fonctions qui permettent aux programmes d'interagir avec le SE. Les programmes peuvent appeler les fonctions de ces bibliothèques. Le SE charge et fait le lien (link) entre les fonctions des librairies et le programme d'une application. Le *linker* s'organise pour charger les librairies en RAM et de faire en sorte que les fonctions puissent fonctionner peu importe où elles ont été placées.
  - **Un ensemble d'outils système:** Des petits programmes qui permettent d'exécuter des fonctions du SE. Par exemple, changer la protection de certains fichiers.
  - **Des programmes applications de base:** Un fourre-tout d'applications qui n'ont pas nécessairement à voir avec le SE mais sont fournies avec celui-ci pour permettre de faire des tâches de base (calculatrice, agenda, …).

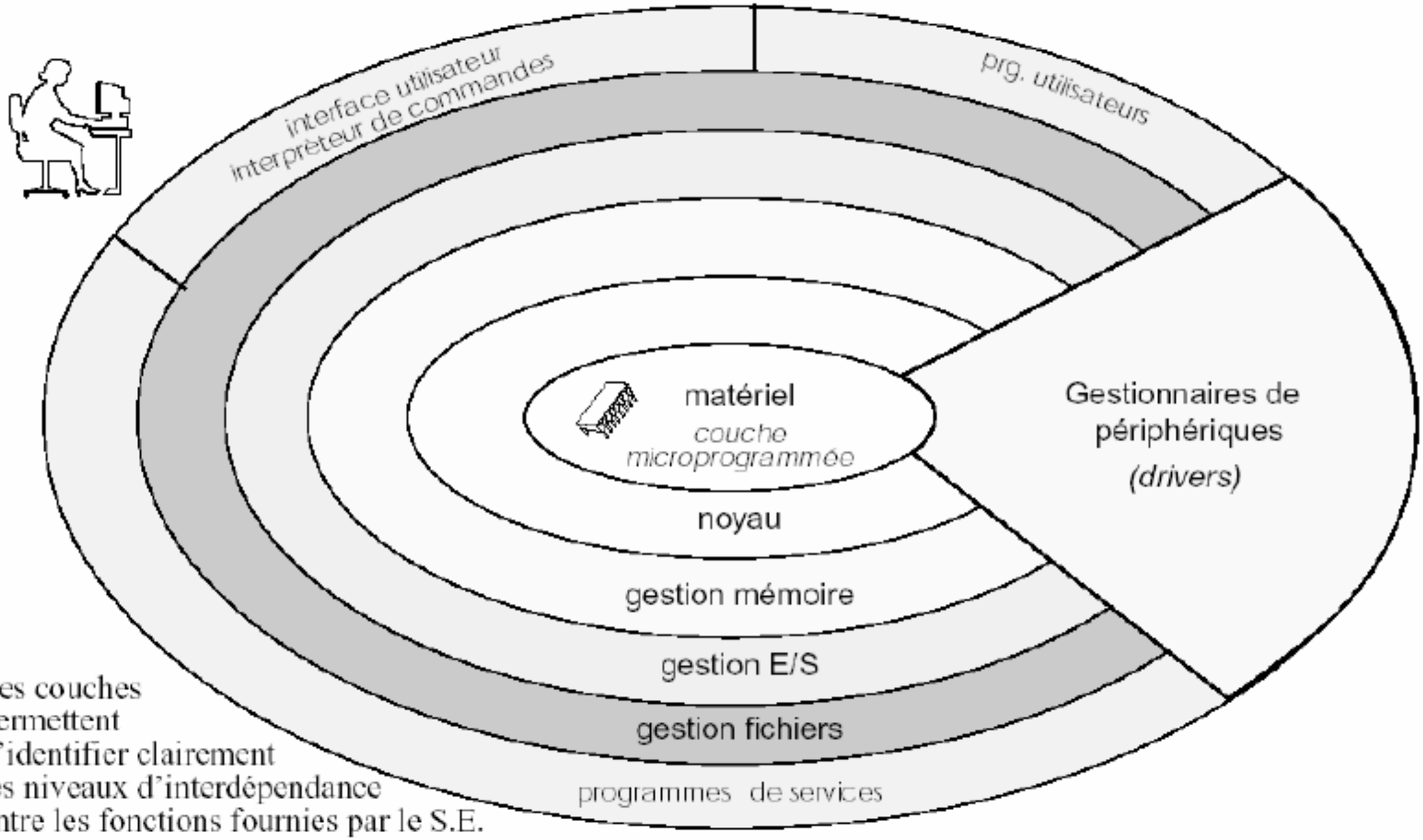
Composition (II)







Composition (III)



Les couches permettent d'identifier clairement les niveaux d'interdépendance entre les fonctions fournies par le S.E.

Le noyau (kernel)

- Le noyau assure les fonctions suivantes:
  - gestion des périphériques par des pilotes.
  - gestion des files d'exécution (i.e. des processus):
    - attribution de la mémoire pour chaque processus,
    - ordonnancement des processus, c-à-d la répartition du temps pour chaque processus sur le ou les processeurs.
    - synchronisation et communication entre les processus. Ceci comprend des services de synchronisation, des échanges de messages et le partage de segment de mémoire entre les processus.
  - gestion des fichiers au moyen d'un système de fichiers.
  - gestion des protocoles réseau comme TCP/IP (internet) et IPX (plutôt pour les réseaux locaux).

Le noyau (II)

- **Boot:** Le programme qui initialise l'ordinateur. Permet aux premiers services d'accéder aux applications système: gestion de la mémoire, aux disques durs et aux périphériques.
- **Répartition de la mémoire centrale (RAM):**
  - réserve des portions mémoire pour le SE. Aucune autre application ne peut accéder à cette mémoire. Assure que le SE ne se fasse pas modifier par des applications.
  - réserve des portions de mémoire pour les applications.
- **Assure une certaine indépendance entre les applications et le matériel:** À travers les fonctions du noyau, il est possible d'interagir avec une clé USB sans en savoir plus (la sorte, la quantité de mémoire). La clé USB se comporte alors comme un disque dur…

Bibliothèques dynamiques

- Elles sont nommées *librairies* en anglais.
- Elles regroupent les opérations du SE souvent utilisées par les applications. Exemple : accès aux fichiers, accès aux périphériques d'E/S, accès à la mémoire RAM, etc…
- Ces fonctions sont disponibles aux programmes applicatifs. Ces programmes peuvent maintenant interagir avec les fonctions du SE.

Le système de fichiers

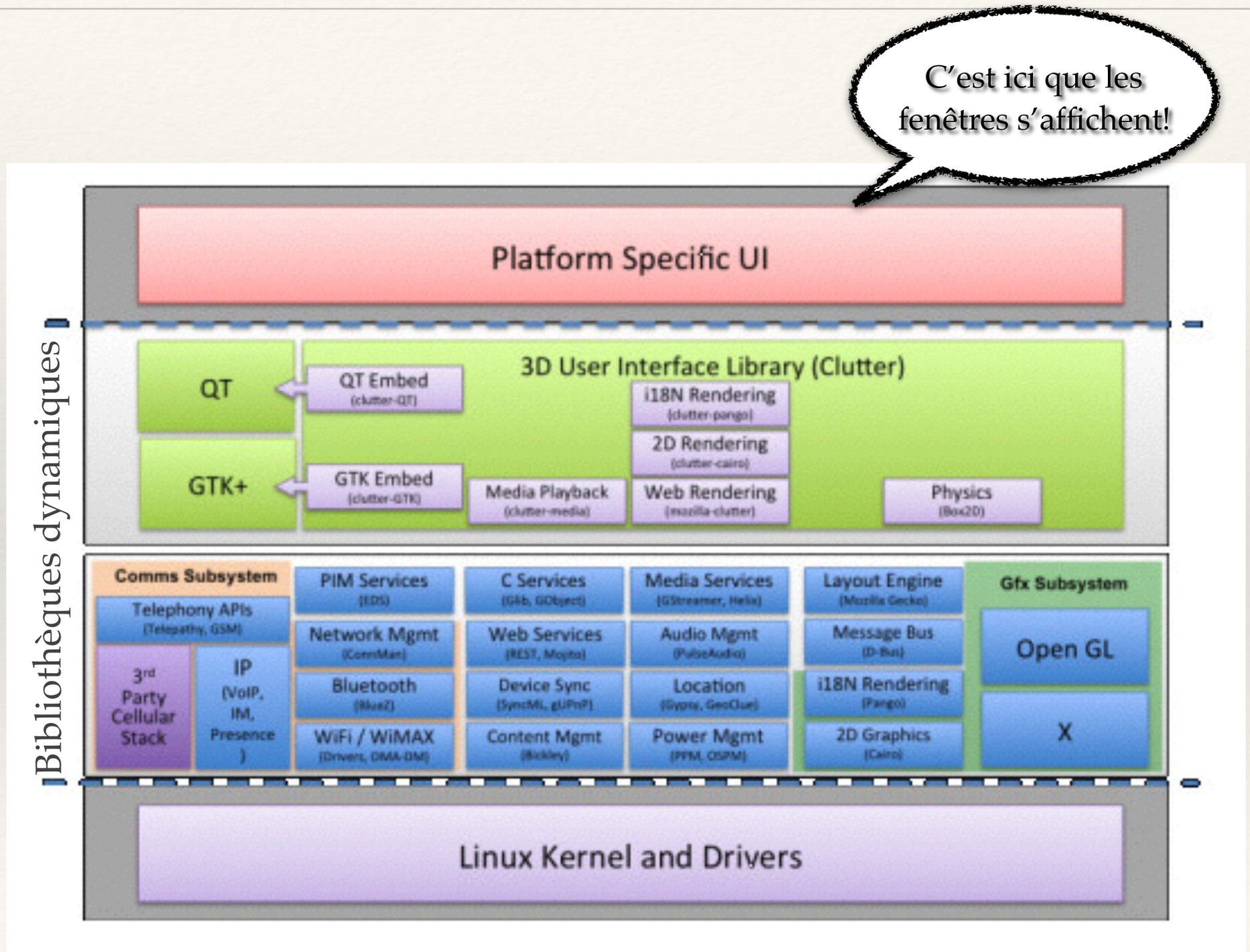
c:
\Documents\toto.txt

- Le SE met en place une structure de données permettant de ranger les données et les organiser sous forme de fichier sur des mémoires secondaires (e.g. disques durs, disquettes, CD-ROM, DVD, clé USB, etc…).
- L'organisation est la plupart du temps hiérarchique. Par exemple sur MAC OS X: `~/Documents/toto.txt` indique que `toto.txt` est un fichier de l'usager courant dans le répertoire `Documents`.
- Ce stockage est persistant. Il demeure présent jusqu'au moment ou l'utilisateur en décide autrement.
- Une telle gestion des fichiers permet de conserver une grande quantité de données qui peut être partagée entre les programmes.
- Permet à l'usager d'avoir une vue abstraite de ses données. Les données accessibles en suivant un chemin d'accès: `~salvail/Documents/`.

Interface homme-machine

- Des utilitaires permettent d'interagir avec le SE:
  - UNIX, LINUX, MAC OS: les Shell (coquille par opposition à noyau),
  - PC: MSDOS.
- Interfaces graphiques:
  - Xterm, MAC OS, Windows, LINUX-KDE, LINUX-KUBUNTU, etc…

Architecture en couches LINUX



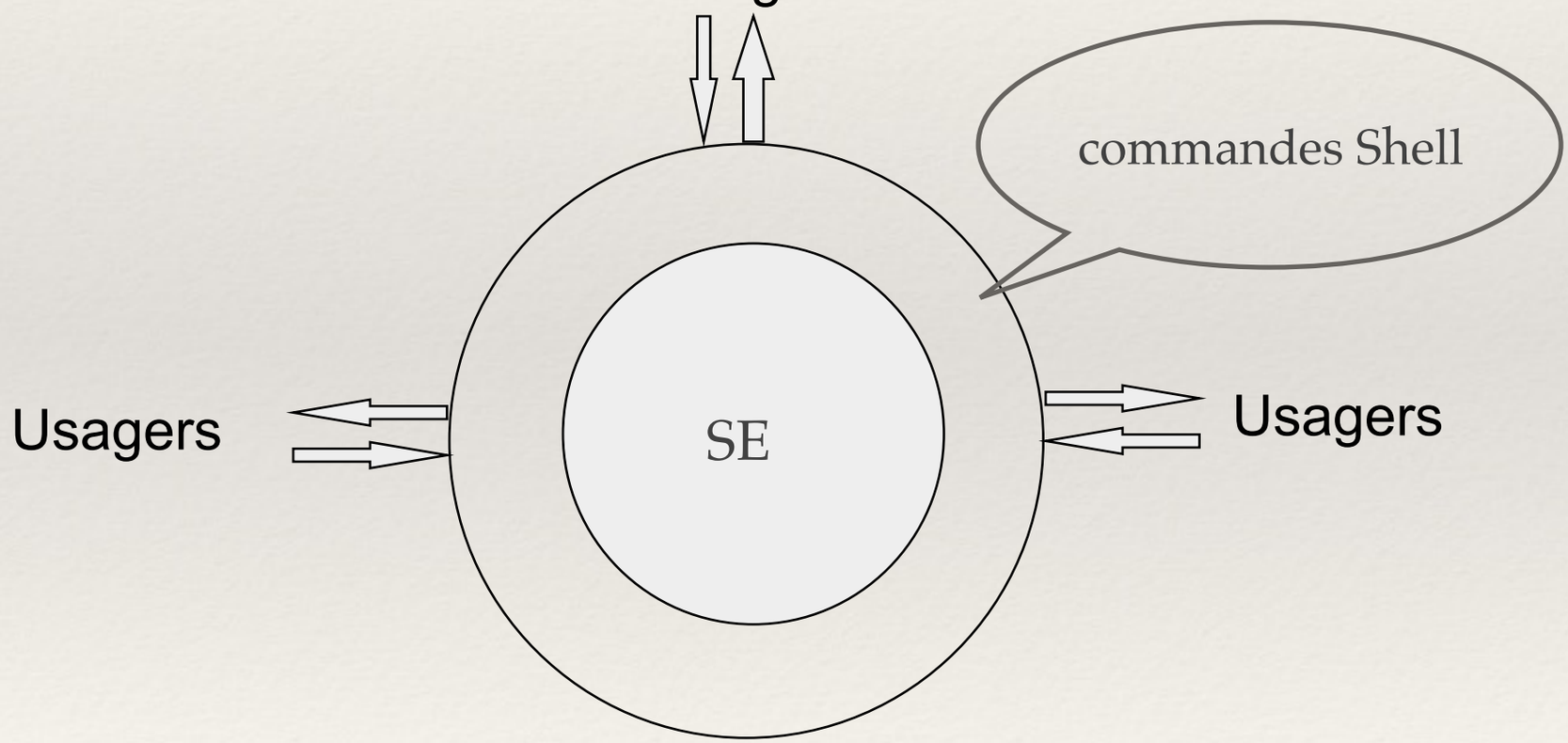
Shell

- Les SEs qui sont dérivés de UNIX: LINUX, MAC OS ont une application appelée Shell qui permet d'avoir accès à ses fonction par des commandes. Un Shell est un interpréteur de commandes du SE.
- Un Shell permet à l'utilisateur de piloter des périphériques en ignorant les caractéristiques du matériel qu'il utilise, de la gestion des adresses physiques, etc…
- Permet de faire de l'administration système: créer des usagers, donner des permissions aux usagers pour accéder aux ressources…

Exemples de commandes Shell

- `ls` : donnes la liste des fichiers dans le répertoire courant. L'option `-all` donne tous les détails des fichiers: `ls -all`.
  - En général, on peut restreindre les fichiers affichés en filtrant: `ls -all toto*` donne tous les fichiers dont les noms débutent par `toto`.
- `mv ~/toto.txt ~/tata.txt` renomme `toto.txt` en `tata.txt` et l'endroit où trouver toto.txt est dans le répertoire principal de l'usager courant.
- `cd ~/Documents/Presentations/` remplace le répertoire courant par `~/Documents/Presentations/`.
- `pwd` donne le répertoire courant avec son chemin d'accès.

L'interface Shell



Multi-utilisateur & multi-tâche

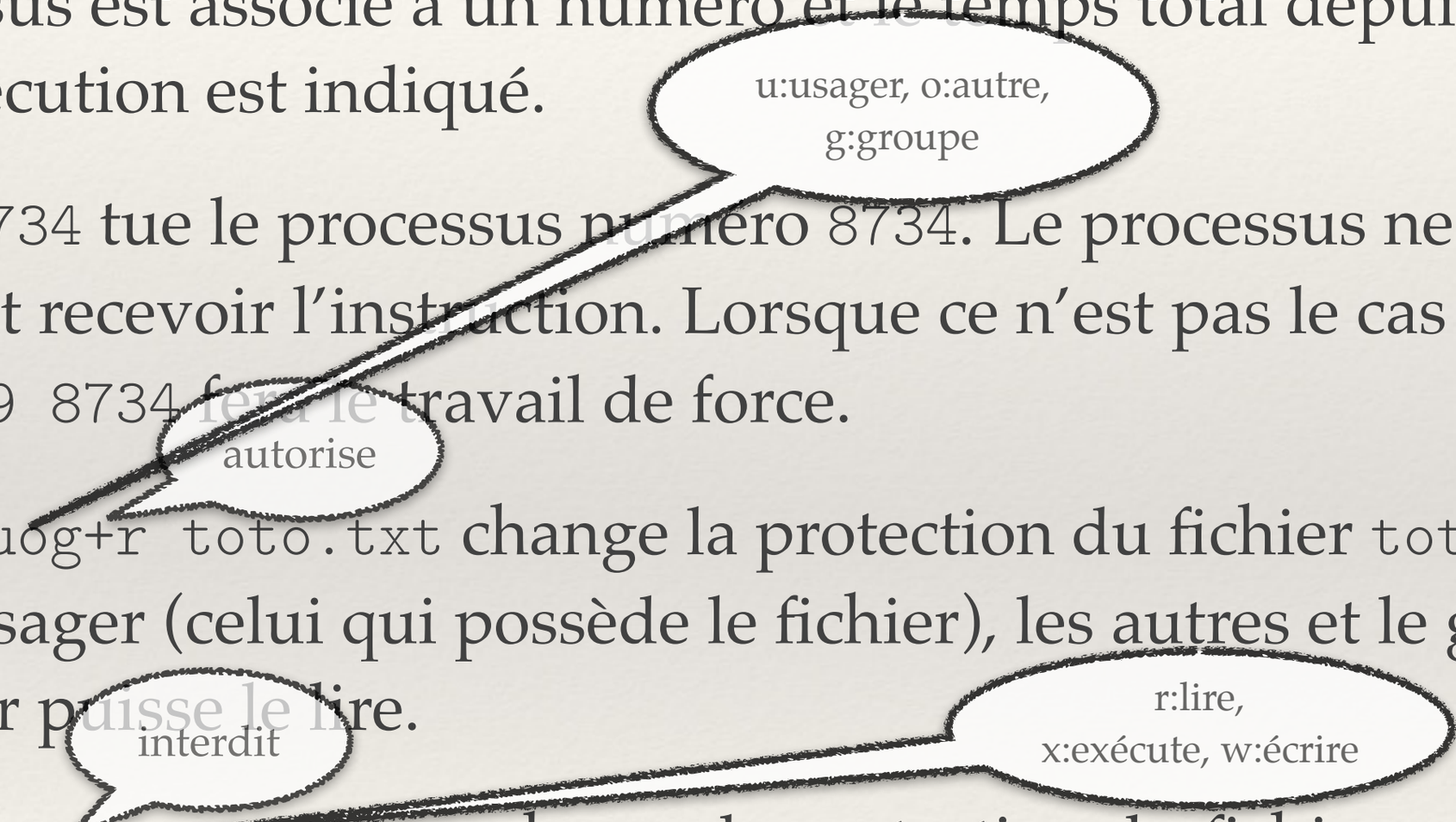
- **Multi-utilisateur:**
  - permet à plusieurs usagers d'utiliser un même ordinateur en même temps.
  - Le SE doit s'assurer que les usagers ne peuvent pas interférer. Il doit également s'assurer qu'un usager ne peut accéder qu'aux fichiers et ressources auxquels il a droit.
- **Multi-tâche:**
  - permet à un usager de lancer plusieurs applications en même temps.
  - Le SE doit partager le temps de calcul entre les applications. Il doit respecter les priorités: pendant qu'un programme fonctionne il doit être possible de l'interrompre lorsqu'un événement prioritaire survient. Par exemple, les déplacements de souris, sélection de fenêtre.
  - Le SE doit également être équitable dans le temps alloué aux applications pour qu'elles puissent rouler efficacement.

Type de multi-tâche

- Un SE qui ne permet pas le multi-tâche est dit *mono-tâche*.
- Un SE qui permet un multi-tâche simple où chaque tâche est responsable pour permettre à une autre tâche de s'exécuter (cette autorisation doit être explicite) est dit *multi-tâche coopératif*.
- Un SE qui a la capacité de stopper et d'exécuter une tâche planifiée en cours au profit d'une autre tâche de priorité supérieure est dit *multi-tâche préemptif*.

Les commandes UNIX pour la gestion des tâches et des utilisateurs

- `ps -u salvail` donne la liste des processus de l'usager `salvail`. Un processus est associé à un numéro et le temps total depuis le début de son exécution est indiqué.
- `kill 8734` tue le processus numéro `8734`. Le processus ne meure que s'il peut recevoir l'instruction. Lorsque ce n'est pas le cas alors `kill -9 8734` fera le travail de force.
- `chmod uog+r toto.txt` change la protection du fichier `toto.txt` pour que l'usager (celui qui possède le fichier), les autres et le groupe de l'usager puisse le lire.
- `chmod og-rxw toto.bin` change la protection du fichier exécutable `toto.bin` pour que les autres et le groupe de l'usager ne puisse ni lire, ni exécuter ni écrire `toto.bin`.



Évolution des SE

- Au début de l'informatique, les ordinateurs n'avaient pas de SE.
- Au début des années 60, les utilisateurs apportaient leur programme sur cartes aux opérateurs qui les roulaient pour eux et retournaient le résultat le lendemain (une tâche ou *job*). Les opérateurs devaient faire des manipulations significatives pour permettre le changement de tâche.
- Le SE (à l'origine) était un programme qui aidait les opérateurs dans leur travail consistant à préparer les tâches.
- Les opérateurs s'occupaient de partager le temps de calcul entre les tâches des usagers.

Évolution des SE (II)

- Les usagers ne pouvaient pas interagir avec l'ordinateur durant l'exécution de leur programme.
- SE pour les processus interactifs:
  - Permet au programme de dialoguer avec l'usager via un terminal ou une station de travail.
  - La tache pouvait maintenant être exécutée en temps réel.
  - Les usagers pouvaient maintenant obtenir les réponses rapidement
  - Les machines étaient trop grosses et dispendieuses pour servir qu'un seul usager.
  - Il devint commun que plusieurs usagers demandent des services interactifs en même temps.

Évolution des SE (III)

- SE pour partager le temps de calcul:
  - Pour servir plusieurs usagers en temps réel, le SE exécutait les programmes en partageant le temps d'exécution.
  - Chaque tâche avait une tranche de temps prédéterminée.
  - À la fin de la tranche de temps, la job courante était mise de côté pour qu'une autre puisse démarrer.
  - En passant d'une tâche à l'autre rapidement, le SE créait l'illusion de plusieurs tâches exécutées simultanément.

Évolution des SE (IV)

- Sans trancher le temps, un ordinateur passe la majorité de son temps à attendre que quelque chose arrive: qu'un périphérique annonce un événement ou un usager demande l'exécution d'une nouvelle tâche.
- Dans ce contexte, un ensemble de tâches est exécuté plus rapidement en tranchant le temps…
- C'est devenu monnaie courante…

Plusieurs SE sur une même machine?

- Il est possible d'avoir plusieurs SE sur une même machine.
  - Chacun peut être utilisé lorsque voulu par l'usager.
- Il n'est pas possible cependant de rouler deux SE en même temps sur une même machine.
- Exemple: LINUX+Windows sur un PC, LINUX+MAC OS X sur un Mac, VMS+UNIX sur un VAX (un type de PDP).

Le gestionnaire d'horaire (scheduler)

- Conserve un enregistrement des tâches qui sont présentes, ajoute de nouvelles tâches, retire les tâches complétées:
  - Indique la mémoire assignées à une tâche,
  - La priorité d'une tâche et
  - Son statut pour l'exécution: prêt, en attente.

L’ordonnanceur de tâches

- Assure que les tâches prêtes à être exécutées le sont.
- Le temps de calcul du CPU est divisé en segments de courtes durées (50ms). Nous appelons ces segments des tranches de temps.
- Lorsque la tranche de temps d’une tâche est terminée, l’ordonnanceur permet au gestionnaire d’horaire de mettre à jour l’état du processus qui correspond à chaque tâche. Il sélectionne ensuite la prochaine tâche à exécuter.

Comparaisons des SE

Nom	Année d'origine	mono/multi-utilisateur	mono/multi-tâche
UNIX (grosses machines)	1972	multi	préemptif
DOS	1981	mono	mono
Mac OS	1984	mono	mono & coopératif
Win 3.1	1992	mono	coopératif
Win 95	1995	mono	coopératif
Mac OS X	1999	multi	préemptif
Win NT/2000	2000	multi	préemptif
Win XP	2001	multi	préemptif

Historique de UNIX