Éléments d'informatique – Cours 2. Éléments de systèmes d'exploitation

Pierre Fouilhoux et Lucas Létocart

This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

- Éléments d'architecture des ordinateurs
- Éléments de systèmes d'exploitation
- Programmation structurée impérative (éléments de langage C)
  - Structure d'un programme C
  - Variables : déclaration (et initialisation), affectaction
  - Évaluation d'expressions
  - Instructions de contrôle : if, for, while
  - Types de données : entiers, caractères, réels, tableaux, enregistrements
  - Fonctions d'entrées/sorties (scanf/printf)
  - Écriture et appel de fonctions
  - Débogage
- Notions de compilation
  - Analyse lexicale, analyse syntaxique, analyse sémantique
  - préprocesseur du compilateur C (include, define)
  - Édition de lien
- Algorithmes élémentaires (+ initiation à la complexité)
- Méthodologie de résolution, manipulation sous linux

Le(s) système(s) d'exploitation

- Abstraction du matériel
- Gestionnaire de processus
- Environnement de travail

Démos

Machine virtuelle Linux

Divers

Programmation

- Structure d'un programme C
- Variables impératives et traduction de l'affectation

Le(s) système(s) d'exploitation

Il n'y a pas de définition précise.

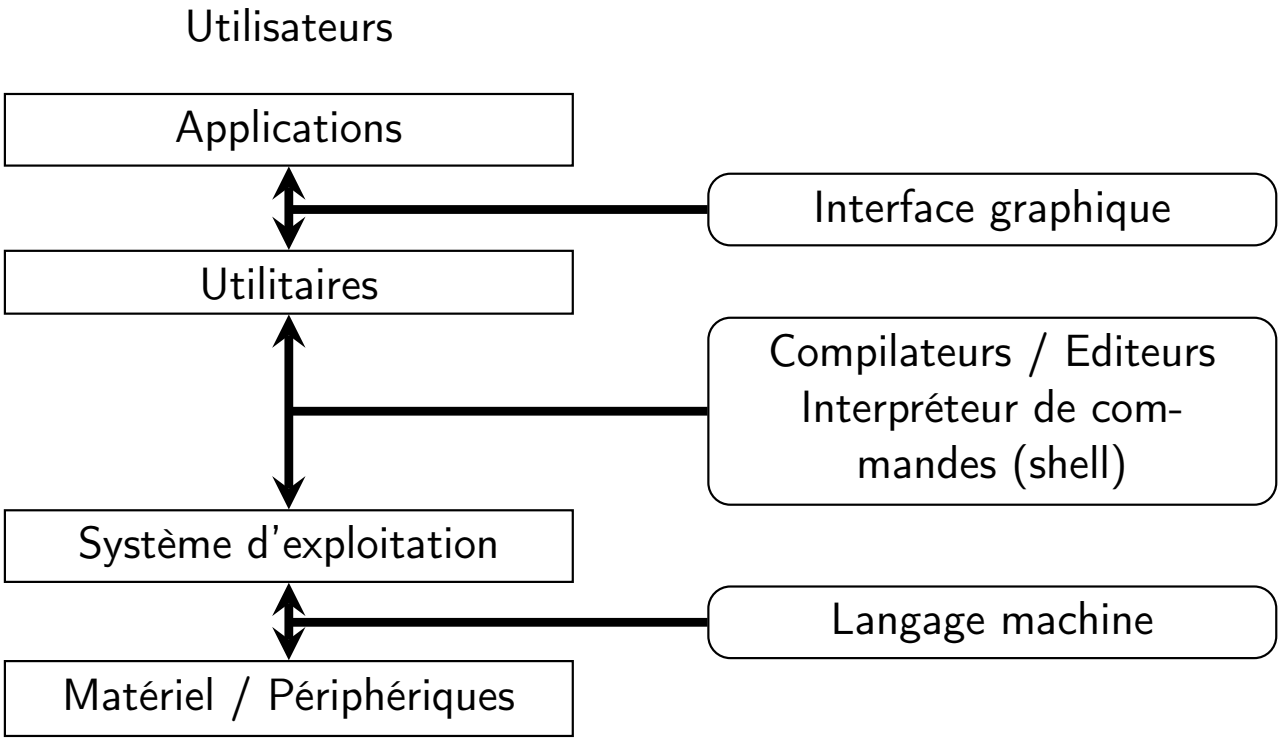
Definition (système d'exploitation)

Un système d'exploitation est un ensemble de programmes qui servent d'interface entre le matériel et les logiciels applicatifs.

Nous distinguerons trois rôles endossés par le SE :

- Pilotage et abstraction du matériel
- Gestionnaire et ordonnanceur de processus
- Environnement de travail

Organisation en couches



Abstraction du matériel

Un système d'exploitation contient un ensemble de programmes dont le rôle est de piloter les périphériques matériels.
Le système d'exploitation sert d'intermédiaire entre le matériel et les autres applications pour :

- simplifier l'utilisation du matériel
- réglementer et protéger l'accès au matériel (intégrité)
- donner corps aux abstractions utiles (le système de fichiers, ou la notion d'utilisateur).

Au tout début de la séquence de démarrage, le programme d'amorçage ou *bootstrap*, un programme résidant dans un mémoire spécifique, déclenche le chargement du système d'exploitation en mémoire.

Gestionnaire de processus

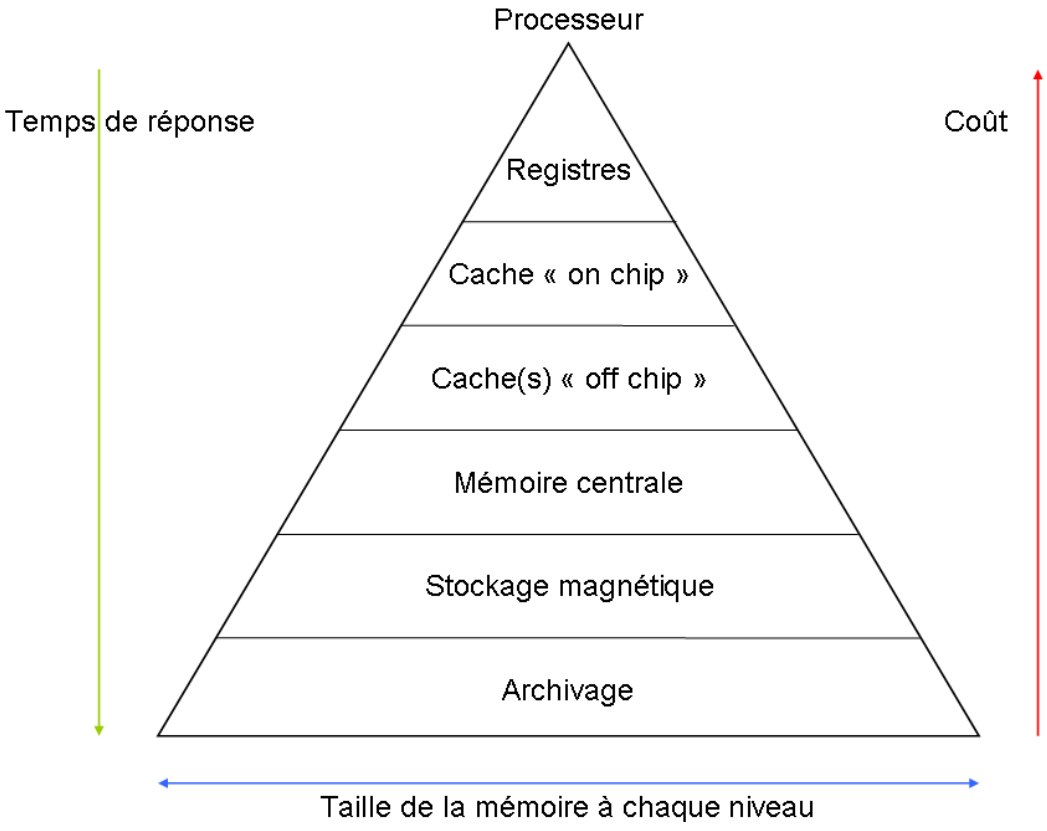
On appelle processus un programme informatique en cours d'exécution.
Le système d'exploitation assure la gestion des processus.
Notamment :

- Le chargement des programmes à partir d'une mémoire de masse vers la mémoire de travail (et saut sur le point d'entrée du programme).
- Le partage du temps processeur.
- Le partage de la mémoire.
- Le système gère également les interruptions matérielles, la communication entre les processus etc.

Temps partagé, ordonnancement

- Sur un système multitâches, il y a un seul processeur (ou un petit nombre de processeurs) et de nombreux processus.
- Chaque processus est exécuté tour à tour, pendant un intervalle de temps court, de l'ordre de 10 à 100 ms, après quoi il retourne en attente (jusqu'à ce que son tour revienne).
- Ceci donne une illusion de simultanéité et de parallélisme.
- Le temps partagé peut être coopératif (c'est le programme du processus qui prévoit de rendre temporairement le processeur), ou préemptif (le processus est interrompu par le système).
- Algorithme d'ordonnancement. C'est le système d'exploitation qui fixe l'ordre dans lequel ces processus sont exécutés.
- Exemple d'ordonnancement : le tourniquet.
- Un peu de temps est perdu à la commutation de contexte.

Mémoire partagée, mémoire virtuelle



Mémoire partagée, mémoire virtuelle

- La mémoire est en général virtualisée.
- Cela signifie que les applications n'ont pas d'accès direct à la mémoire physique. Les adresses mémoires manipulées par les processus ne sont pas directement les adresses physiques, elles sont virtuelles.
- Un calcul d'adresse est effectué par le système pour passer des adresses virtuelles aux adresses physiques et réciproquement.
- Ceci permet de faire comme si chaque programme était seul résidant en mémoire et simplifie le travail de programmation.
- Certaines portions inactives de mémoire de travail peuvent également être stockée sur une mémoire de masse (disque dur), de manière à étendre la quantité de mémoire de travail disponible (attention aux temps d'accès !).

Environnement de travail

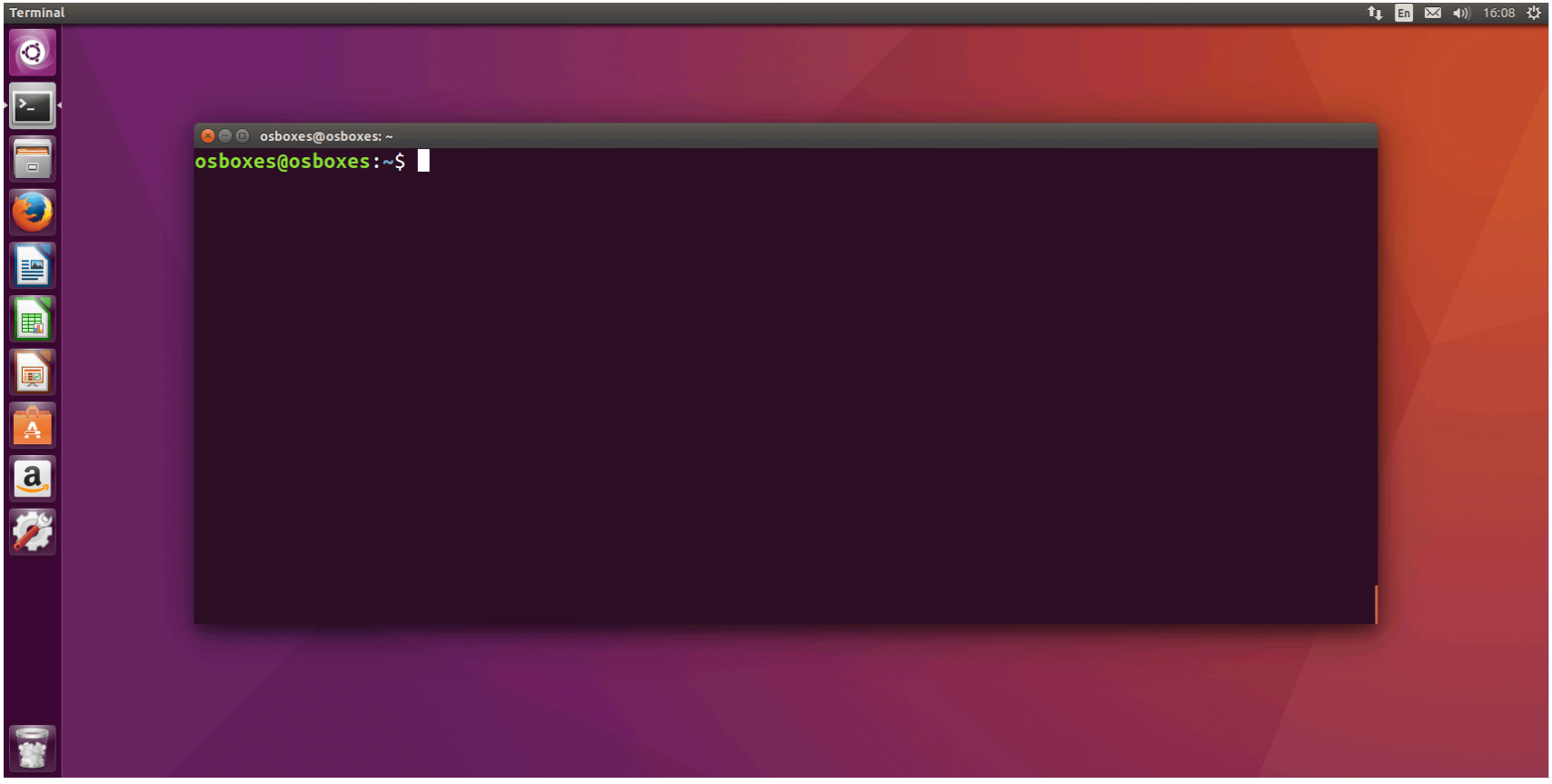
Selon un point de vue extensif, le système est vu comme un ensemble de logiciels (navigateur web, éditeur de texte, etc.) qui accompagnent le matériel et sont installés en même temps que le système d'exploitation proprement dit.

- Environnement graphique (fenêtres, bureau, corbeille)
- Services inter-applications (correction orthographique, annuaire, etc.)
- Multi-utilisateurs.

Le terminal

- Interface de communication avec le système
  couplée à un interpréteur de commande
- Exécution de programme à travers le terminal (commandes systèmes, programmes écrits en TP, etc.)
  $\rightarrow$ A privilégier en TP
- L'utilisateur a la possibilité
  - d'interroger le système
  - d'interagir avec le système
  - d'écrire des programmes (scripts)
- Principe d'utilisation
  1. l'utilisateur tape des commandes sous forme de texte
  2. l'interpréteur évalue le texte pour identifier les commandes et leur arguments
  3. l'interpréteur exécute les commandes

Le terminal



Démos

Machine virtuelle Linux

Didacticiel disponible :
`http://www.math.univ-paris13.fr/~chaussar/Divers/Didacticiel/UbuntuVirtuel/installer_xubuntu.htm`

Démo : Installation et utilisation dans VirtualBox

Divers

- Les TD et les TP suivront une méthodologie de résolution à appliquer (faire des exemples, écrire un algorithme, traduire en langage C, tester).
- Pour vous entraîner à programmer en langage C sur un ordinateur personnel, il y a par exemple la solution `codeblocks`.
- Sous Windows 10, on peut désormais activer un noyau linux.
- Alternative plus difficile : virtual box et une ubuntu.
  Didacticiel disponible :
  `http://www.math.univ-paris13.fr/~chaussar/Divers/Didacticiel/UbuntuVirtuel/installer_xubuntu.htm`

La programmation structurée

Definition (Programmation structurée)

Programmer par *blocs* d'instructions en combinant ces blocs de trois manières :

1. exécuter les blocs les uns à la suite des autres (*séquence*)
2. si une certaine condition est vraie, exécuter un bloc sinon en exécuter un autre (*sélection*)
3. recommencer l'exécution d'un bloc tant qu'une certaine condition est vraie (*répétition*).

Un bloc peut lui-même contenir une combinaison de blocs.

Cette idée simple conduisit à l'introduction de langages dits de haut niveau tel le langage C.
Aujourd'hui nous allons voir la sélection en langage C, le `if else`, et une première forme de répétition en C, le `for`

Structure d'un programme C

Code source

```c
/* Declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */

/* Declaration des constantes et types utilisateurs */

/* Declaration des fonctions utilisateurs */

/* Fonction principale */
int main()
{
    /* Declaration et initialisation des variables */
    ...
    /* valeur fonction */
    return EXIT_SUCCESS;
}

/* Definitions des fonctions utilisateurs */
```

Les commentaires sont ignorés lors de la traduction en langage machine.

Traduction de l'affectation

Code source

```
...
/* Fonction principale */
int main()
{
    /* Declaration et
    int x = 5;


    x = 3 * x + 1;
    ...
```

Schéma de traduction