Systèmes d'exploitation

William SCHMITT

2018-2019

1 Introduction

1.1 Bibliographie

Tanenbaum : Modern Operating Systems Silberschatz : Operating Systems Concepts

1.2 Plan

Cours/TD
- Moniteurs
- Sémaphores
- futex

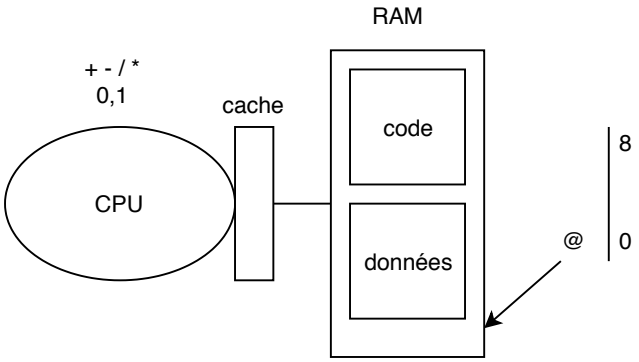
TP (pas à rendre)
- 1ère période : rappels de C
- 2ème période : mmap

TP (à rendre)
- Allocateur mémoire virtuel (malloc) : révisions sur les pointeurs
- Shell (fork, exec, redirection IO)
- Thread (lecteur vidéo multithread)

Présentation scientifique : article de usenix.org (de 2019, 2018 ou dans les *best papers* des 3 dernières années) à présenter devant la classe.

1.3 Examen

- QCM : 1h (20 à 25 questions)



— TP :
  — 2h (75%)
  — Présentation scientifique (25%)

Notation : 50/50.

1.4 Fibonacci

Comparaison avec/sans printf, ratio de performance : 397

Avec redirection de stdout dans /dev/null, le ratio plonge à 31.

Lorsque les tampons sont pleins (dans le terminal), on bloque les producteurs de données le temps de vider les tampons (et donc d'afficher les résultats).

Facteurs décorrélés de l'algorithme permettant des gains :
— IO
— Gestion de la mémoire

2 Hardware

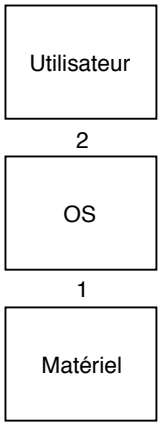
On se base sur un modèle simpliste.

Depuis les années 1990, la vitesse du CPU a dépassé celle de la mémoire. En un cycle CPU (pour un CPU à 4GHz), un photon peut parcourir 7,5cm. On rajoute à cette époque du cache dans les CPU pour compenser la lenteur de la mémoire.

De plus, la taille de la mémoire a explosé.

3 Système d'exploitation

Dans les premier ordinateurs, on réglait l'état mémoire à la main avec des interrupteurs.



Un OS est à la base composé de différentes routines qui ont deux buts :

1. Gestion du matériel (partage, arbitrage, partition, droits/utilisateurs)
   — Processeurs (dont la complexité augmente)
   — Disques/SSD/Burst buffer/Bandes magnétiques
   — RAM
   — Interfaces réseau
2. Abstractions
   — Fichiers, répertoires, systèmes de fichiers
   — processus
   — utilisateurs
   — socket réseau

3.1 Composition d'un OS

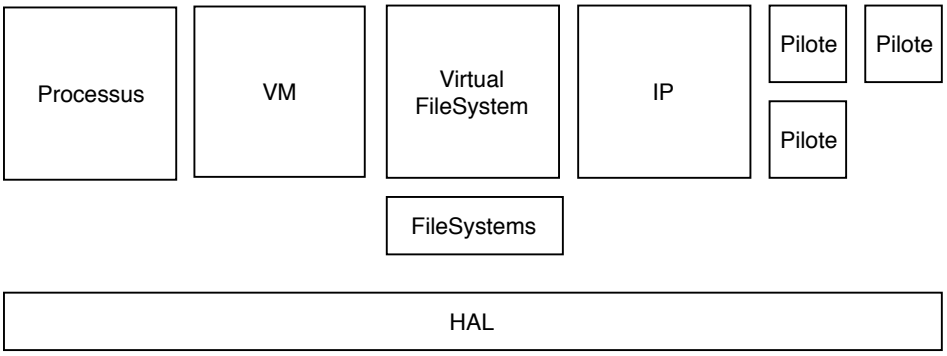
Le noyau Linux est constitué d'environ 18 millions de lignes en C. Ingérable par une seule personne, il est décomposé en modules exposant des API puis compilé en un programme comme les autres, à quelques exceptions près.

Mode d'exécution un OS s'exécute dans un mode spécial et a accès à des instructions spéciales.

Interruptions et exceptions permettent de passer du mode user au mode kernel. La différence entre les deux : les interruptions sont masquables.

Bibliothèque système côté utilisateur permettant (notamment) d'éviter de coder les interruptions à la main.

S'il s'agit d'un programme comme les autres : où et quand est-il en RAM ? Où et quand s'exécute-t-il ?



Il est toujours en RAM (notamment pour pouvoir répondre aux interruptions), avec certaines nuances : tout l'OS n'est pas en mémoire ! Pour Linux, seuls certains modules sont chargés, notamment pour des drivers. Sous Windows, les parties non utilisées sont sur disque (swap).

En termes d'exécution, il n'est exécuté que lors d'interruptions/exceptions.

Déroulement du boot
- Intel ME (management engine), codé sur le processeur et peut faire tourner Minix.
- BIOS/UEFI : exécuté directement par le CPU au boot, accessible comme de la mémoire normale, pour chercher le chargeur et le mettre en mémoire. Fourni par le fabricant de mobo, il y a déjà des interactions avec le matériel avant le chargement de l'OS. L'accroissement de complexité de l'UEFI (possibilité de surfer) présente des problématiques de sécurité.
- Chargeur : mise en RAM des données sur disque
  - Windows
  - GRUB
  - Autre...
- Initialisation
- Vecteurs d'interruption/exception

La maîtrise de la machine physique devient de plus en plus lointaine, même au niveau des OS.

3.2 Modularité