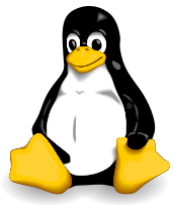
M1101 : Introduction aux Systèmes d'Exploitation (OS - Operating Systems)

Guillaume Urvoy-Keller

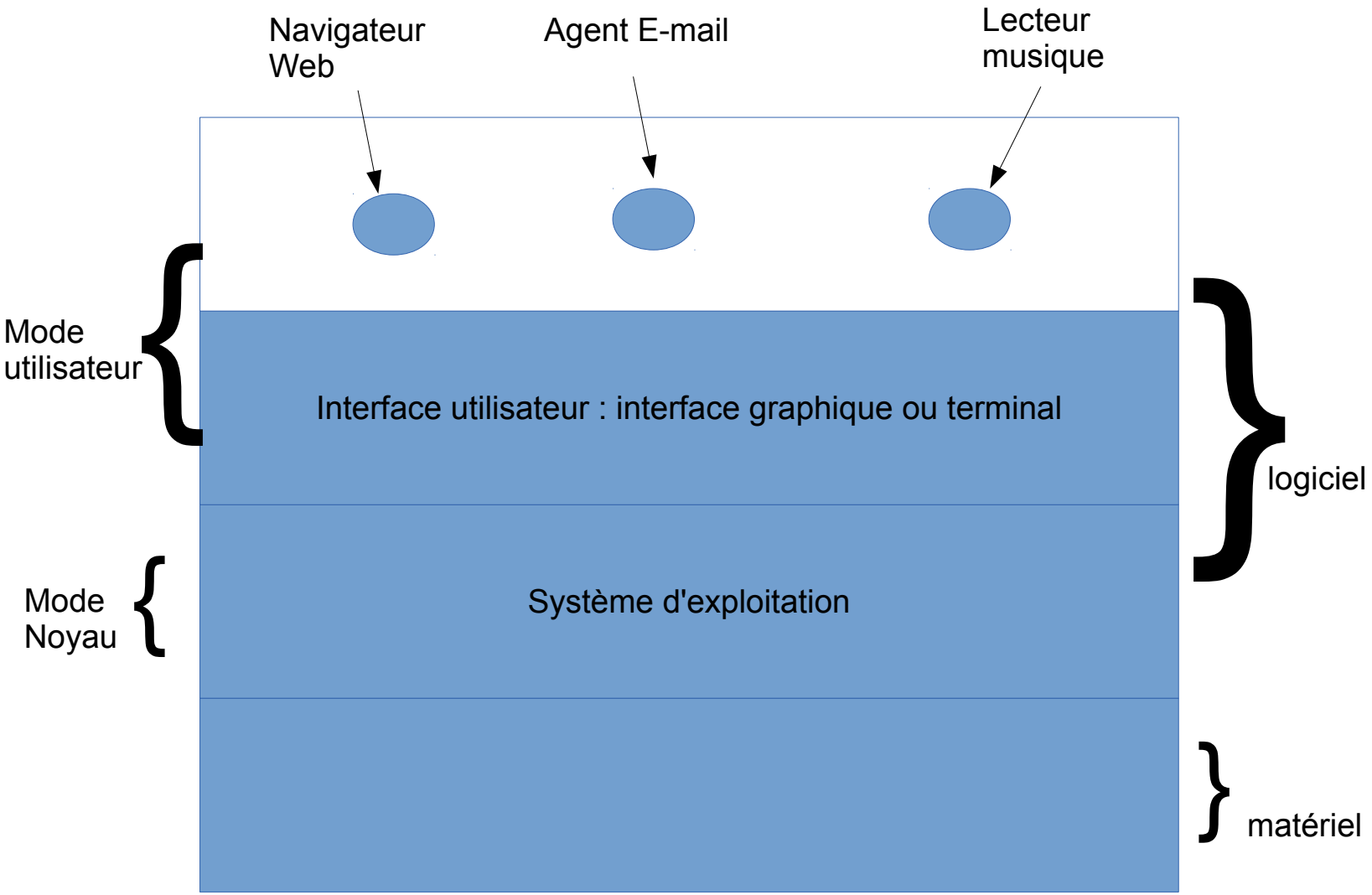
Source : A. Tannenbaum "Modern Operating Systems", Pearson



L'ordinateur moderne : un système complexe

- Si tous les programmeurs devaient comprendre complètement comment fonctionne un tel système $\rightarrow$ aucun programme ne serait jamais écrit!
- Gérer tous les composants (disque, écran, clé USB, carte réseau) de façon optimale est complexe $\rightarrow$ il faut une couche intermédiaire entre l'utilisateur/programmeur et le matériel!

The big Picture



Les modes de fonctionnement (du processeur)

Mode Noyau

- Mode de fonctionnement de l'OS
- A tous les droits sur le matériel et l'exécution de toutes les instructions machines (= instructions que le processeur comprend)

Mode Utilisateur

- Mode de fonctionnement de tous les autres programmes (ex: Firefox, skype, etc.)
- Pas d'accès direct au matériel
- Un sous-ensemble des instructions du processeur

Interface utilisateur/OS

Interface utilisateur

- Le niveau le plus bas des programmes utilisateurs
- Utilisé pour démarrer les autres programmes : navigateur, client mél (mail)
- Interface graphique ou textuelle

OS

- Temps de vie bien supérieur à celle des programmes (10 ans ou plus)
- Très gros : Linux ou Windows font + de 5 millions de lignes de code

Les missions de l'OS

Mission n°1: Fournir au programmeur une abstraction simple des resources disponibles : disque, mémoire, réseau

Mission n°2 : Gérer ces resources.

L'OS: une interface de haut-niveau pour le programmeur

Architecture d'un ordinateur au niveau le plus bas (langage machine) est complexe à maitriser, surtout pour les périphériques d'E/S (entrées/sorties : disques, écran, clavier, etc...)

Exemple (obsolète)

- Lecteur de disquette PD765 a 16 commandes pour lire, écrire, bouger le bras, rédémarrer son moteur, avec des codes de retour contenant 23 états codés dans 7 octets!!!!!
- Le programmeur veut une **interface abstraite** : le disque doit être vu comme un certain nombre de fichiers que l'on peut : ouvrir, lire, écrire, fermer.

L'OS: une interface de haut-niveau pour le programmeur - suite

- L'OS est là pour fournir les bonnes abstractions : cacher les détails au programmeur
- Dans l'exemple précédent, on veut la même interface pour un vieux lecteur de disquette, un disque SSD (Solid State Drive) ou un disque réseau.

L'OS: un gestionnaire de resources

Les resources

- Le processeur (CPU : Central Processing Unit)
- La mémoire vive
- les disques
- l'accès à la carte réseau

Exemple

- 2 programmes veulent simultanément envoyer un fichier à l'imprimante
- Si ils accédent directement : 1 ligne du premier, 1 ligne du second...
- Solution : L'OS stocke les fichiers à imprimer dans des buffers

L'OS: un gestionnaire de resources - suite

Mot-clef

Multiplexage : plusieurs programmes/utilisateurs se partagent des resources SANS en avoir conscience (sans devoir s'organiser)

Mulitplexage temporel

Imprimante, CPU (un seul programme s'éxécute à chaque instant), disque

Mulitplexage spatial

Mémoire (plusieurs programmes)

Historique ordinateurs/OS

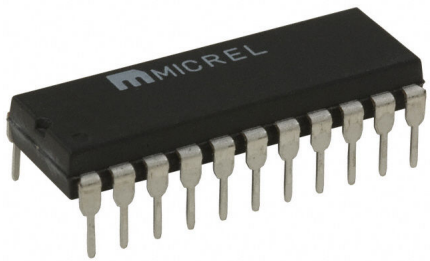
Première génération (1945-1955)

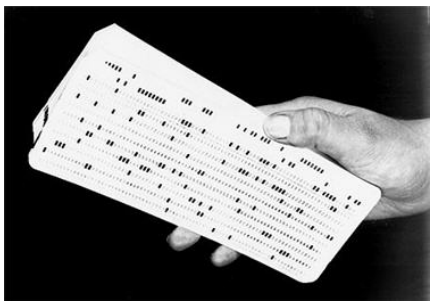
- A base de tubes à vide
- Ex: ENIAC (Université de Pensylvannie)

Historique ordinateurs/OS

Seconde génération (1955-1965)

- Transistors remplacent les tubes à vide
- Grandes entreprises comme IBM entrent en jeu
- Séparation entre développeurs et concepteurs
- Apparation des systèmes d'exploitation.
- Un seul utilisateur à la fois. Son programme est écrit sur des cartes perforées.
- Langage fortran ou assembleur (langage machine = jeu d'instruction du processeur)





Historique ordinateurs/OS

Troisième génération (1965-1980)

- Circuits programmés (puces) remplacent les tansistors. Miniaturisation.
- Notion de **multi-programmation** :
  - plusieurs programmes actifs. Lorsqu'un programme est en attente d'une entrée sortie et que la CPU est libre, un second occupe la CPU
  - Possibles car la mémoire est divisée entre les programmes et le système d'exploitation.
- Notion de temps partagé
  - Plusieurs (centaines?) utilisateurs, chacun avec un terminal physique qui accède à l'ordinateur central
  - MIT, Bell Labs et General Electrics developpent MULTICS
- Multics sera adapté à un mini-ordinateur $\rightarrow$ UNIX

Historique ordinateurs/OS

D'Unix à Linux

- Code source UNIX a été largement distribué
- Différentes versions développées : System V (AT&T), BSD (Berkeley Software Distribution)
- IEEE a normalisé une librairie de programmation standard, POSIX, qui permet de compiler un programme sur tous les système Unix
- Une version pour l'éducation : MINIX
- En 1994, un étudiant de l'Université d'Helsinki, Linus Torvald, développe son propre OS, basé sur les idées d'UNIX et compatible POSIX: LINUX!!

Historique ordinateurs/OS

Quatrième générations (1980 -)

- Les PCs (Personal Computer)
- Miniaturisation de l'électronique $\rightarrow$ mini-ordinateur deviennent des micro-ordinateurs
- En 1980, IBM a conçu le PC et cherchait un système d'exploitation et des programmes autour du processeur Intel 8080
- Bill Gates, qui a conçu le langage de programmation BASIC entre en jeu
- Il trouve une petite entreprise de Seattle qui a un système d'exploitation : DOS (Disk Operating System)
- Il leur achète et fonde une société (Microsoft) avec un employé, Tim Paterson, qui finalise MS-DOS (MS comme Microsoft)
- MS-DOS intial primitif, mais incorpore, petit à petit, des caractéristiques d'Unix

Historique ordinateurs/OS

Mainframe (gauche) vs. mini-ordinateur (milieu) vs. PC (droite)



Historique ordinateurs/OS

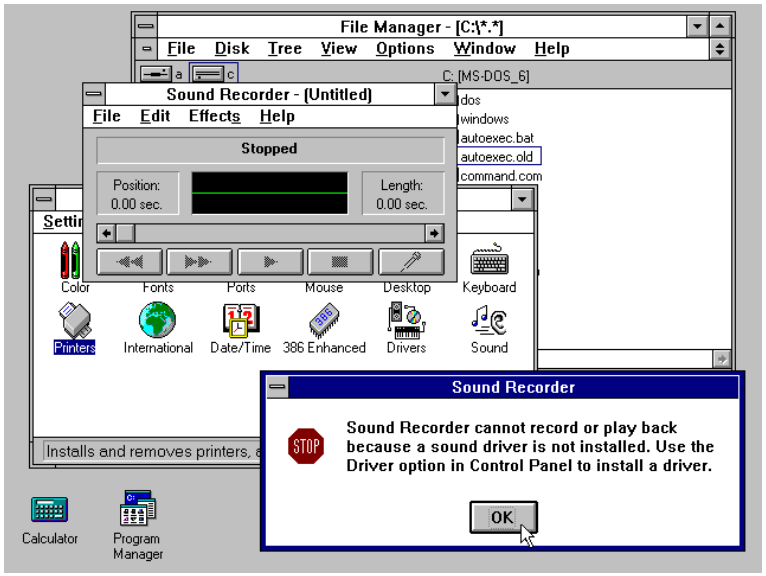
PC vs. Mac

- Steve Jobs, co-inventeur d'Apple, visite un jour la société Xérox et voit une interface graphique (prototype)
- Il développe un système d'exploitation et le matériel qui va avec : Lisa (un flop) puis le Macintosh (un succès)
- Evolutions de MS-DOS influencées par le Macintosh avec l'ajout d'une interface graphique

Historique ordinateurs/OS

Evolutions Windows

- De 1985 à 1995, Windows était une couche au dessus de MS-DOS
- Windows 95 et 98 : MS-DOS disparait petit à petit. Toujours en architecture 16 bits
- Windows NT: 32 bits
- 2001-2007 : Windows XP. Stable
- 2007 : Vista puis Windows 7, 8



Historique ordinateurs/OS

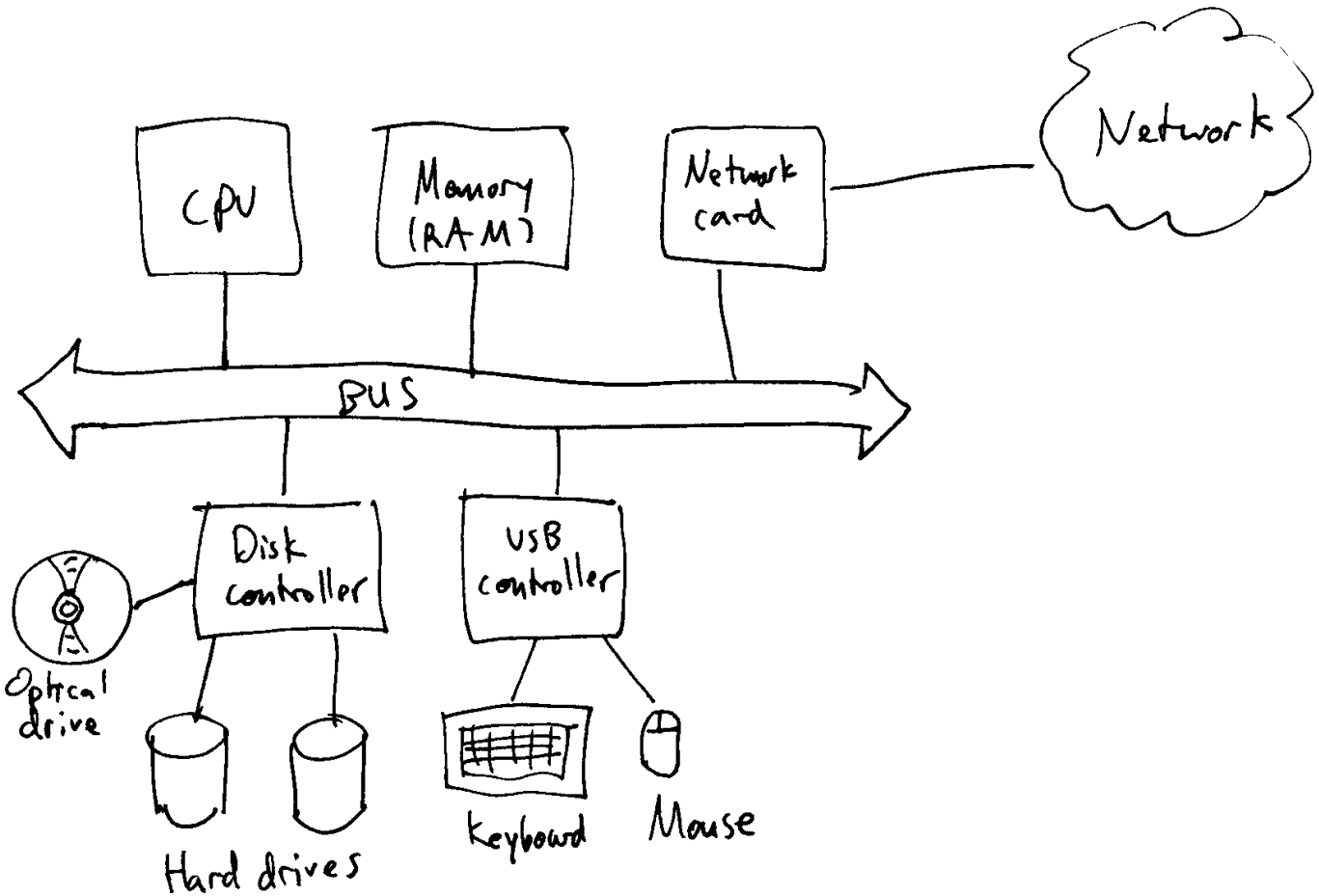
Evolutions Unix

- FreeBsD à partir de BSD. Une version modifiée est la base de MacOSX
- Interface graphique : X Windows System (X11)
- Interfaces modernes comme Gnome ou KDE au dessus de X11

Revue rapide des composants matériels

Pourquoi?

Car l'OS doit connaitre tous les détails du matériel pour bien fonctionner



Revue rapide des composants matériels

La CPU

- Prend desinstructions de la mémoire et les exécute
- Chaque CPU a son jeu propre d'instruction. Ex: x86 Intel
  `http://en.wikipedia.org/wiki/X86_instruction_listings`
- La mémoire centrale (RAM) est lente $\rightarrow$ registres dans la CPU où sont stockées les opérandes (= données sur lesquelles on calcule)
- Registre : mémoire ultra-rapide dans le processeur (mais toute petite...)
- En plus des registres pour les calculs, on a des registres spéciaux, notamment :
  - PC (Program Counter) : adresse de la prochaine instruction à exécuter
  - PSW (Program Status Word) : plusieurs bits et notamment le mode (noyau ou utilisateur)

Revue rapide des composants matériels

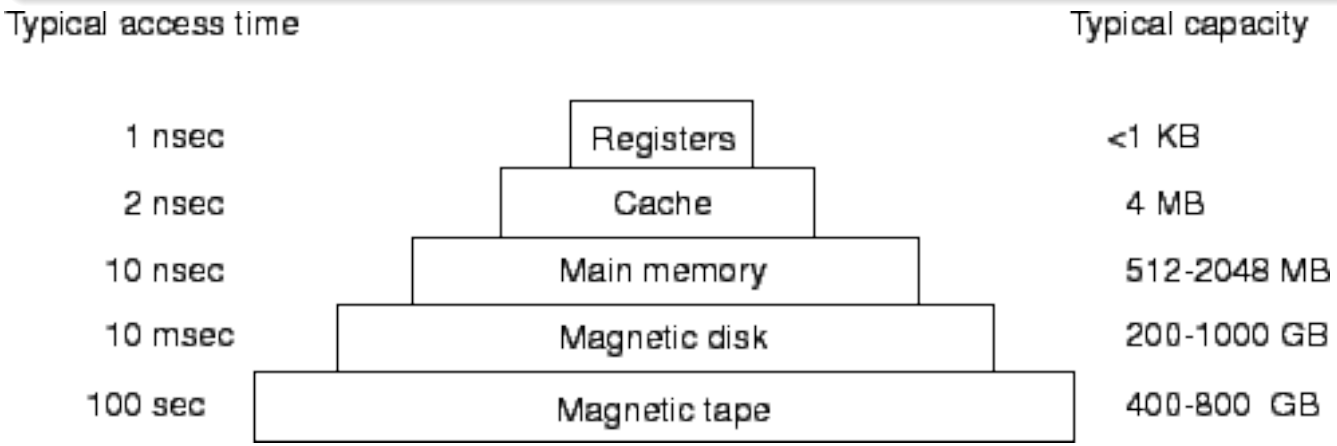
La CPU - suite

- Chaque fois qu'un programme est suspendu (pour éxecuter plusieurs en parallèle), la CPU doit stocker l'état de tous les registres.
- En mode utilisateur (vos programmes, ex: firefox, thunderbird, etc), on a pas accès à la totalité des instructions.
  - Notamment toutes les entrées/sorties : disque, mémoire, réseau
  - Pourquoi? C'est l'OS en mode noyau qui assure la répartition équitable des resources.
- Il faut une interface pour demander au noyau d'exécuter ces instructions pour le compte du programme : notion d'**appels système**.

Revue rapide des composants matériels

La mémoire

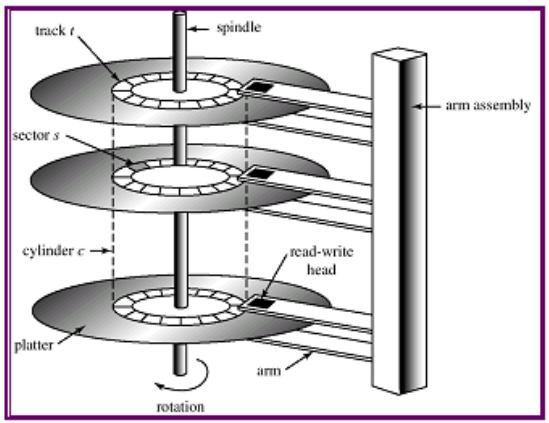
- RAM (main memory sur figure)
- MMU (Memory Management Unit) : les programmes ont l'impression de pouvoir utiliser toute la mémoire. Les adresses (virtuelles) qu'ils utilisent sont transformées en adresses physiques par une partie de la CPU, la MMU.
- Principe du cache : stocker des bouts d'une grande quantité d'information (ex : fichier) dans un système physique plus rapide car on pense que ce bout va être accéder prochainement



Revue rapide des composants matériels

Disque dur

- Deux ordres de grandeur (x100) moins cher que RAM, deux ordres de grandeurs plus grand, trois ordres de grandeur moins rapide
- Chaque disque est divisé en plages. Les plages sur lesquelles sont les bras forment un cylindre. Chaque plage est divisée en secteurs.
- Temps de positonnenment des têtes de lecture élevé
- Vitesse de lecture rapide (50 à 200 Moctets/s)



Revue rapide des composants matériels

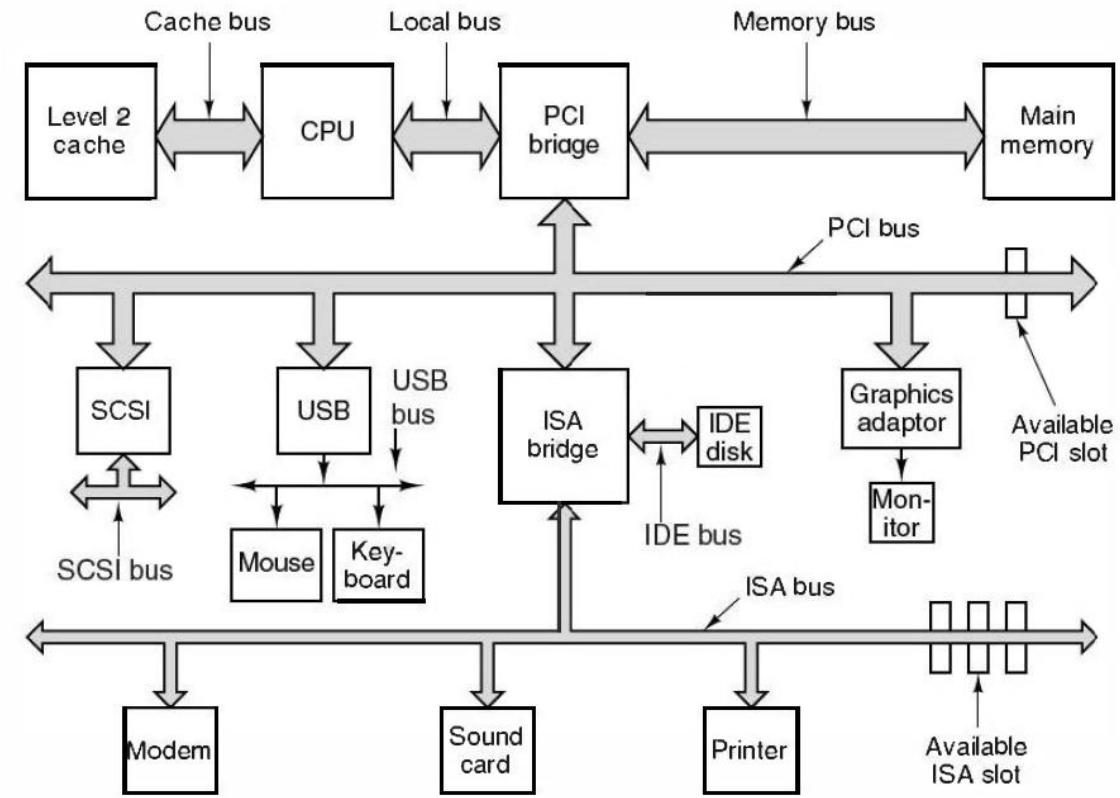
Périphériques E/S

- E/S $\leftrightarrow$ I/O (Input/Output)
- Deux parties en général : un contrôleur et le périphérique lui-même
- Contrôleur agit vis-à-vis de l'OS comme l'OS vis à vis des programmeurs : il fournit une version simplifiée du périphérique.
- Le dialogue OS contrôleur se fait avec un programme spécial : **le pilote (driver)**
  - Un pilote par OS (un pout Windows, un pour Linux....)
  - Fonctionne en mode noyau.

Revue rapide des composants matériels

Bus

- Initiallement, un unique bus
- Maintenant plusieurs, à différentes vitesses en pratiques.
- Ex: PCI à 528 Moctets/s et ISA à 16.6 Moctet/s (ISA n'est plus utilisé)



The structure of a large Pentium system

Gestion des E/S

Périphériques vis-à-vis de l'OS

- Périphériques **très** lents par rapport à la CPU
- Classiquement, la CPU envoie la commande (ex: lecture d'un secteur d'un disque) au contrôleur et traite un autre programme
- Quand le contrôleur a les données, il envoie une **interruption** à la CPU pour dire qu'il est prêt
- Notion de DMA (Direct Memory Access) : on alloue de la mémoire au contrôleur qui copie lui même les données.

Programmes utilisateurs vis-à-vis des périphériques

- Pas d'accès direct : médiation de l'OS
- Utilisation d'un appel système : le programme donne la main à l'OS en disant ce qu'il veut (ex : accès disque, envoie données sur carte réseau)

Boot Machine

BIOS

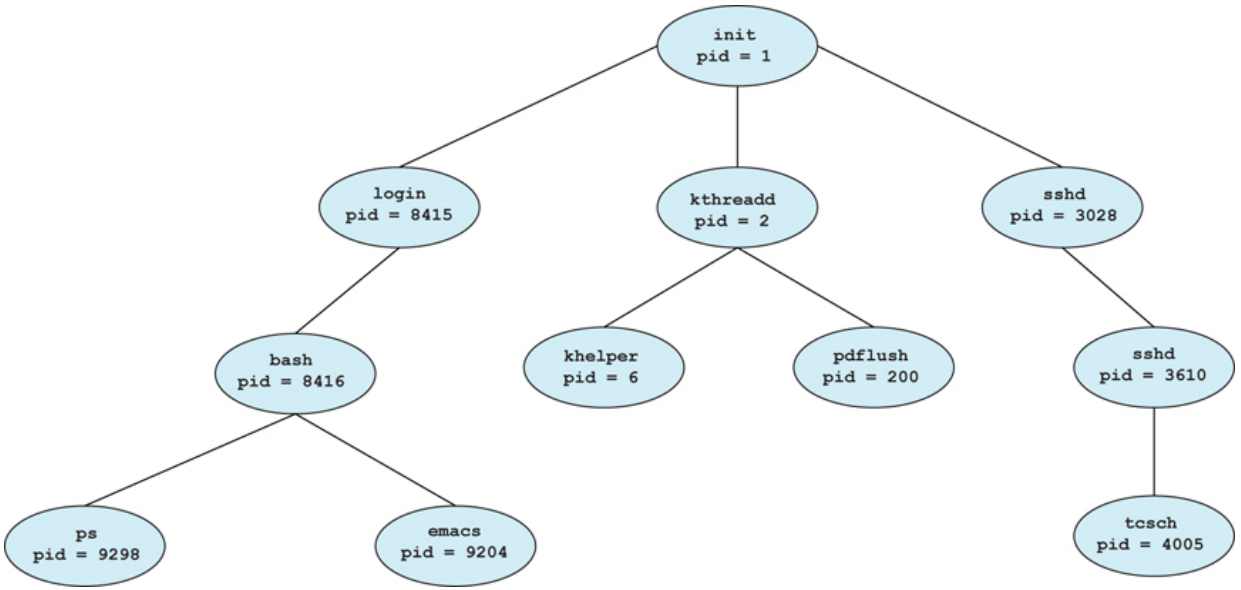
- Basic Input Output Interface
- Programme dans mémoire RAM non volatile qui permet
  - d'activer l'écran, le clavier,
  - scanner les périphériques
  - et lancer le démarrage de l'OS sur le bon périphériques (CDROM, disque, réseau)
  - OS interroge BIOS pour apprendre quels sont les périphériques.

Abstractions clefs des OS

- Les processus
- L'espace d'adressage
- Les fichiers
- (il y en a d'autres)

Abstractions clefs des OS : les processus

- processus = programme en train de s'exécuter. Comprend :
  - le programme exécutable
  - Les données
  - Un état des registres, ex: PC (Program Counter)
  - Les fichiers ouverts, etc
- Un processus est créé toujours par un autre processus $\rightarrow$ arbre des processus
- un processus est lié à un utilisateur (voir commande top)



Abstractions clefs des OS : les processus

Commande top (sous Unix) liste les processus : les utilisateurs ont un numéro associé (UID) ; les processus aussi (PID)

urvoy@kheops: ~/www/Cours/Tel — top — 151×24

urvoy@kheo...s/Tel — top | bash | bash | bash

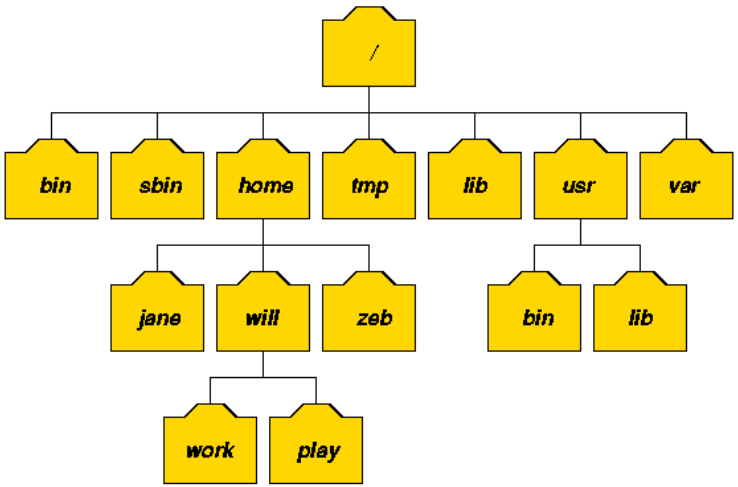
```
Processes: 77 total, 3 running, 74 sleeping, 435 threads
Load Avg: 1.19, 1.21, 1.17  CPU usage: 10.0% user, 13.75% sys, 76.25% idle  SharedLibs: 3084K resident, 5988K data,
MemRegions: 57775 total, 1205M resident, 17M private, 833M shared. PhysMem: 666M wired, 1660M active, 783M inactive,
VM: 171G vsize, 1043M framework vsize, 2430811(0) pageins, 618496(0) pageouts.  Networks: packets: 2717504/1722M in,
Disks: 1570765/29G read, 880267/50G written.

PID    COMMAND      %CPU TIME     #TH  #WQ  #POR #MREGS RPRVT  RSHRD  RSIZE  VPRVT  VSIZE  PGRP  PPID STATE    UID
10074  screencaptur 0.2  00:00.05 2    1    41-  81-   472K-  8188K  2812K- 13M-   2664M- 190   1    sleeping 501
10072  top          16.7 00:01.60 1/1  0    25   33    1604K  264K   2184K  17M    2378M  10072 272  running  0
10047- VBoxSVC      0.0  00:03.11 22   1    141  168   5888K  11M    8676K  37M    896M   10046 1    sleeping 501
10045- VBoxXPCOMIPC 0.0  00:01.33 1    0    19   41    1308K  2212K  1800K  19M    591M   10044 1    sleeping 501
10042- VirtualBox   0.0  00:06.67 5    1    116  306   7908K  26M    24M    30M    993M   10042 185  sleeping 501
9816   mdworker     0.0  00:08.06 3    1    50   139   4376K  3864K  30M    106M   2486M  9816  1    sleeping 501
9587   bash         0.0  00:00.01 1    0    17   25    124K   764K   556K   17M    2378M  9587  9586 sleeping 501
9586   login        0.0  00:00.01 1    0    22   53    112K   312K   896K   19M    2379M  9586  269  sleeping 0
9563   bash         0.0  00:00.05 1    0    17   25    32K    764K   568K   17M    2378M  9563  9562 sleeping 501
9562   login        0.0  00:00.06 1    0    22   53    120K   312K   896K   19M    2379M  9562  269  sleeping 0
9547   VDCAssistant 0.0  00:00.17 4    1    91   76    504K   3844K  5696K  31M    2664M  9547  185  sleeping 501
9545-  Skype        0.2  35:29.03 27   2    527  839   73M    71M    83M    148M   1147M  9545  185  sleeping 501
8644-  VPNClient    0.1  00:39.17 2    1    78   217   2440K  15M    8968K  32M    955M   8644  185  sleeping 501
8631   bash         0.0  00:00.04 1    0    17   25    104K   764K   512K   17M    2378M  8631  8629 sleeping 501
8629   login        0.0  00:00.30 1    0    22   53    276K   312K   852K   19M    2379M  8629  269  sleeping 0
8557   eapolclient  0.0  00:00.50 3    1    44   69    724K   2456K  1868K  13M    2404M  8557  13   sleeping 501
8411-  TeXworks     0.0  05:18.58 6    1    113  314   25M    19M    47M    57M    997M   8411  185  sleeping 501
```

Abstractions clefs des OS : l'espace d'addressage

- RAM stocke les processus en cours d'exécution
- Les adresses sont sur 32 ou 64 bits $\rightarrow 2^{32} = 4Go\ 2^{64} \sim 1.810^{7}\ To$
- $\Rightarrow$ Plus d'adresses que de de RAM
- Solution si le processus utilise trop de mémoire : stocker partie utilisée en RAM, le reste sur le disque
- Important : les processus voient les 2^{32} ou 2^{64} adresses comme si ils étaient seuls.
  - C'est la MMU qui multiplexe derrière la scène

Abstractions clefs des OS : les fichiers

- L'OS cache les détails du disque (nombre de plages/secteurs, format FAT32, NTFS, NFS, etc) au travers d'une abstraction : les fichiers (et répertoires)
- **Appels système** pour la création, écriture, lecture, etc
- Les répertoires forment une hiérarchie : un arbre.
- Chemin en Unix avec des / et en Windows avec \ : */Faculty/Prof.Brown/exam.pdf* ou \ Faculty\ Prof.Brown\ exam.pdf



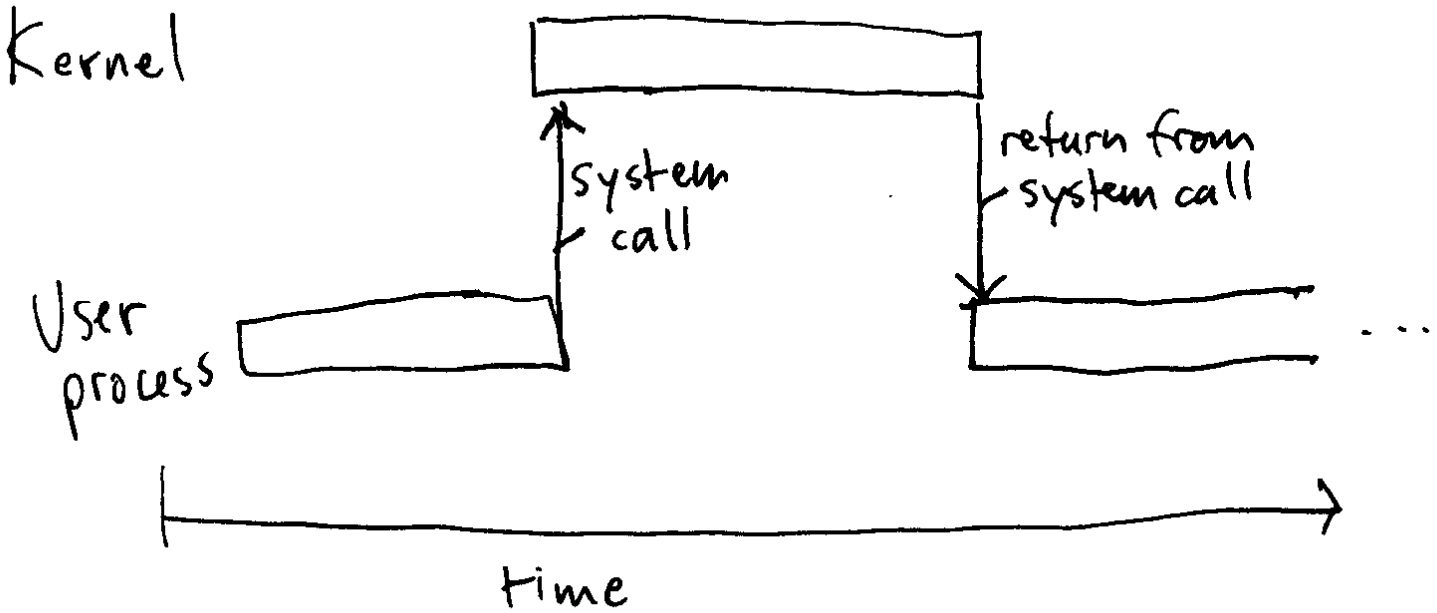
Les appels système

un processus applicatif qui veut lire un N octets dans un fichier

- Règle : à un instant donné, un seul processus occupe le processeur
- **Etape 1 :**
  - Le processus applicatif écrit dans sa pile d'éxecution et les registres les paramètres de l'appel read :
    - un descripteur (=identifiant) de fichier,
    - le nombre d'octets qu'il veut lire,
    - dans quelle variable en mémoire (dans la partie de la mémoire de ce processus!) les données doivent être écrites.
  - Le processus applicatif exécute ensuite l'instruction trap

Les appels système

- **Etape 2:** Il donne ainsi la main à l'OS : la CPU passe en mode noyau
  - L'OS trouve les informations laissées par le processus applicatif
  - Il les exécute et redonne la main (la CPU repasse en mode utilisateur)



L'OS c'est aussi...

De nombreux algorithmes

- Décider quel est le prochain processus qui aura acccès à la CPU.
- Pendant combien de temps?
- Comment faire en sorte que la CPU soit utilisée 100% du temps
- Gérer les retours des périphériques quand ceux-ci ont fini leur travail et font des interruptions. Ex: arrivées de données sur la carte réseau, copies des données du disque en mémoire
- Gérer la répartition de la mémoire. Ex : bloc 1 du processus 1 correspond à bloc physique 100 et bloc 1 du processus 2 (les 2 processus voient la mémoire comme si il étaient seuls!) correspond au bloc 25.