Introduction à Unix/Linux: Cours 1.

Architecture des Ordinateurs et des Systèmes d'Exploitation.

Michaël Quisquater (Maître de Conférences, UVSQ)

Première partie I

Architecture des ordinateurs et notion de programme informatique

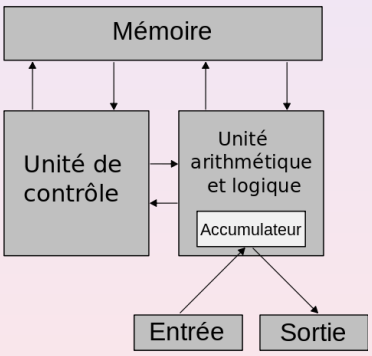
Ordinateur de première génération (1936-1956)

Les ordinateurs numériques à composants "électroniques" sont dits de "première génération".

Exemple :

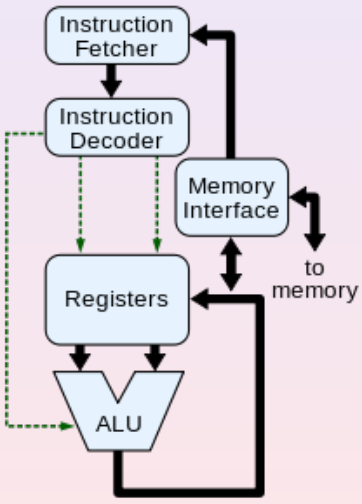
- ENIAC (Electronical Numerical Integrator and Computer)
- Commandé par l'armée américaine pour la balistique.
- Système décimal, Turing-complet
- Non basé sur l'architecture de von Neumann.

Architecture de von Neumann



- Unité arithmétique et logique (UAL ou ALU) : effectue les opérations de base
- Unité de contrôle : chargé du séquençage des opérations
- Mémoire : contient à la fois les données et le programme qui dira à l'unité de contrôle quels calculs à effectuer sur ces données
- Mémoire=mémoire volatile + mémoire permanente
- Entrée-Sortie : permet de communiquer avec le monde extérieur.

Étapes d'exécution d'une instruction



- Fetch : Rechercher instruction
- Decode : décodage de l'instruction (opération et opérandes)
- Execute : Exécution de l'opération
- Writeback : Écriture résultat

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Première génération
Architecture de von Neumann
Seconde génération
Troisième génération
Quatrième génération
Aujourd'hui

Premiers ordinateurs basés sur l'architecture de von Neumann

- 1948 : Small-Scale Experiment Machine (SSEM) première machine basée sur l'architecture de von Neumann. Construite à l'université de Manchester.
- 1951 : UNIVAC I (Universal Automatic Computer). Utilisation de cassettes métalliques au lieu de cartes perforées.
- 1952 : IBM produit l'IBM701 pour la défense américaine.

Ordinateur de seconde génération (1956-1963)

Les ordinateurs de deuxième génération sont basés sur l'invention du transistor (1947).

- 1956 : Ramac (Random Access Method of Accounting and Control) produit par IBM et utilisant un disque dur pour stocker les données.
- 1960 : Digital Equipment Corporation (DEC) lance le PDP-1 (Programmed Data Processor). Concept de mini-ordinateur. Vendu 120.000$ (pas trop cher).

Ordinateur de troisième génération (1963-1971)

Les ordinateurs de troisième génération sont basés sur l'invention du circuit intégré (1958).

- 1963 : un des premiers usages concernait les systèmes embarqués (NASA : guidage d'Apollo et missile balistique LGM-30)
- 1964-1970 : IBM 340 vendu à 14000 exemplaires.
- 1964 : PDP-8, ordinateur non encombrant et destiné aux laboratoires et à la recherche.
- 1967 : le "plan Calcul" destiné à assurer l'indépendance de la France en matière de gros ordinateurs.

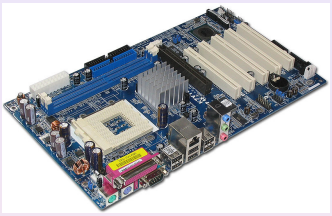
Ordinateur de quatrième génération (1971-fin 1980)

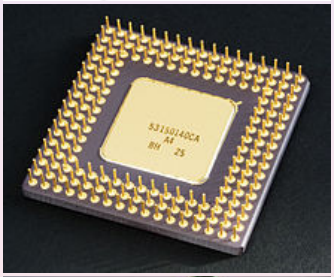
Les ordinateurs de quatrième génération sont basés sur l'invention du microprocesseur.

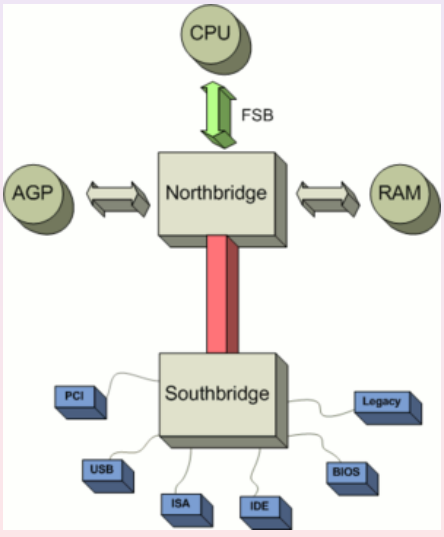
- 1971 : premier microprocesseur commercial, le 4004 de Intel.
- 1973 : premier micro-ordinateur, le Micral N (France, François Gernelle)
- 1974 : processeur Intel 8080 va conduire à la première vague d'ordinateurs personnels.
- 1976 : APPLE I (Steve Wozniak et Steve Jobs)

Depuis lors, les performances n'ont pas cessé de croître mais les principes de construction sont restés très similaires.

Carte mère/CPU/RAM/Cache







Rôle du northbridge

- Une des deux puces du chipset de la carte mère.
- Gère les communications entre le microprocesseur, la RAM, les ports AGP ou PCI express et le southbridge.
- Parfois le northbridge intègre une carte graphique (peu performante mais de faible coût).

Remarques :

- le port AGP (Advance Graphics Port) permet de connecter une carte graphique mais remplacé par PCI express.
- Le port PCI express permet de connecter beaucoup de périphériques et est censé remplacer le PCI (Peripheral Component Interconnect).

Rôle du southbridge

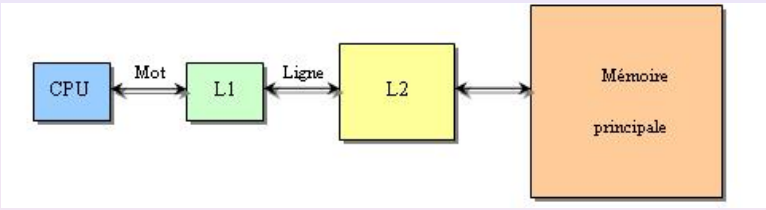
- Une des deux puces du chipset de la carte mère.
- Définit et commande tout ce qui n'est pas géré par le northbridge (PCI, interface PS/2 pour le clavier et la souris, (le port série), (le port parallèle)). Certaines de ces fonctions sont souvent prises en charge par un contrôleur secondaire d'I/O. Dans ce cas, le southbridge fournit une interface à ce contrôleur.
- Il sert aussi de support pour l'interface parallèle ATA (IDE), SATA (disque dur, CDROM), pour l'interface ethernet (réseau), l'USB (Universal Serial Bus) et IEEE1994 (firewire). Parfois le southbridge intègre une carte son.

Évolution

- CPU : Central Processing Unit
- NB : Northbridge
- SB : Southbridge
- GPU : Graphics Processing Unit

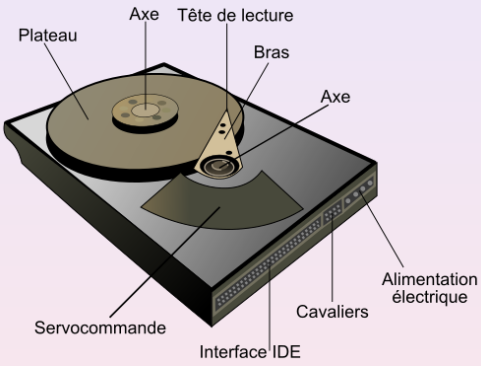
Remarque : Notons que l'architecture Northbridge/Southbridge a tendance à disparaître. Le Northbridge est parfois intégré au processeur ou bien il y a une seule puce pour NB/SB/GPU.

Mémoires physiques



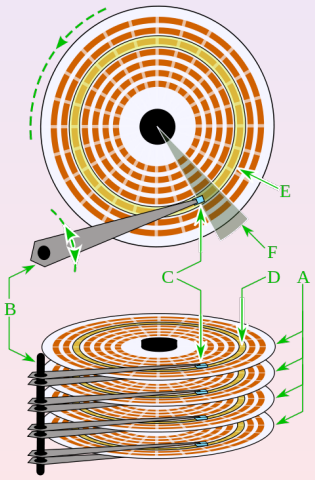
- En pratique, le processeur ne va pas chercher les données ou les instructions immédiatement dans la mémoire RAM.
- Les différentes mémoires par ordre de vitesse (resp. taille) décroissante (resp. croissante) :
  - Registres (intégré dans CPU)
  - Cache de niveau L1 (intégré dans CPU)
  - Cache de niveau L2 (intégré dans CPU)
  - Cache de niveau L3 (intégré dans CPU)
  - Mémoire RAM
  - Disque dur

Composants d'un disque dur



- Plateau(x) tournant et tête(s) qui se déplace(nt)
- Contrôleur (IDE, SATA etc)

Organisation des données



- Chaque plateau (A) : souvent 2 faces utilisables et composé de pistes (E) concentriques séparées les unes des autres.
- Les pistes situées à une même distance de l'axe forme un cylindre (D). Un bras (B) se déplace pour positionner le(s) tête(s) (C) sur celles-ci.
- Les pistes sont découpées en secteurs (F) (souvent 512 ko)

Adresse CHS (Cylinder/Head/Sector) : disque dur et disquette

Adressage CHS

Il faut trois coordonnées pour accéder à un secteur :

- Numéro de la piste qui détermine la position du bras (B).
- Numéro de la tête de lecture (C) (choix du plateau et de la surface).
- Numéro du secteur (bloc).

LBA (Logical Block Adressing) : disque dur et disquette

Adressage LBA

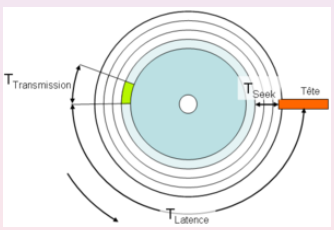
En pratique, le nombre de secteurs n'est plus constant selon les pistes. Le modèle CHS est donc théorique et on lui préfère l'adressage LBA qui consiste à numéroter les secteurs comme si ils formaient un seul ruban.

Description d'un secteur

- Gap : petit blanc entre les secteurs, sert de tampon temporel.
- Servo : Utilisé pour la synchronisation.
- ID : numéro du secteur.
- Données : les données effectives.
- $\sum$: permet détecter (ou corriger) une erreur.

Performance d'un disque dur

Il y a principalement trois caractéristiques à prendre en compte :



- Temps de latence (Latence) : Lié à la vitesse de rotation du disque.
- Temps de recherche (Seek) : Temps mis par la tête pour accéder au cylindre choisi.
- Temps de transfert (Transmission) : Temps mis par les données pour être transférée entre le disque dur et l'ordinateur via l'interface.

Performance d'un disque dur (bis)

Remarque : En pratique, il y a une mémoire vive (cache) dans le contrôleur du disque qui mémorise les blocs suivants le bloc demandé (anticipation). Ce cache permet d'améliorer les performances du disque mais pose des problèmes de synchronisation et de cohérence.

Fiabilité d'un disque dur et formatage bas niveau

- Un disque dur est doté de deux sécurités : le $\sum$ au niveau des secteurs et une table des secteurs défectueux
- Le formatage bas niveau du disque permet d'initialiser la surface de chaque plateau du disque et les pistes qu'ils contiennent. Il sert aussi à inscrire à la racine du disque dur les informations de base concernant le nombre de cylindres, de clusters défectueux, etc. Ce formatage est réalisé en usine en diminuant la sensibilité des têtes en lecture. Cela permet de détecter les blocs défectueux (ou proche de l'être) et de remplir la table en conséquence.
  À réaliser uniquement en cas de problème majeur.

Exemple de processeur (CPU : Central Processing Unit) :

- Marque/modèle : Intel I5
- Fréquence : 2.8 Ghz
- Nombre de coeurs : 4
- taille des caches : L1=4*64Ko, L2=4*256Ko, L3=8Mo (partagé)

Exemple de mémoire RAM :

- Marque(s) : Kingstone ou Corsair
- Capacité : 4Go ou 8 Go parfois plus.

C'est quoi un programme informatique

Programme informatique

- séquence d'instructions qui spécifie étape par étape les opérations à effectuer pour obtenir un résultat
- exprimé sous une forme compréhensible par un ordinateur (éventuellement après une traduction adéquate).

Remarque : En 1842 la comtesse Ada Lovelace crée des diagrammes pour la machine analytique de Babbage, diagrammes qui sont considérés aujourd'hui comme étant les premiers programmes informatiques au monde.

Langage de programmation et notion de compilation d'un programme

- Un ordinateur ne peut exécuter que des programmes binaires (langage machine) qui utilisent des instructions propres au processeur de celui-ci.
- Un programme binaire est difficilement par l'homme.
- Un programme assembleur est la traduction d'un programme binaire en quelque chose d'un peu plus lisible. Exemple : l'instruction machine 10110000 01100001 s'écrit en langage assembleur "movb $0x61,%al". Ce qui signifie "écrire le nombre 97 dans le registre AL". L'assemblage consiste à traduire un programme écrit en assembleur en un programme binaire.

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Notion de programme informatique
Compilation

Langage de programmation et notion de compilation d'un programme

- Afin de faciliter l'écriture de programmes et l'éxécution d'un même programme sur différents types d'ordinateur, des langages de plus hauts niveaux ont été développés.
- Un programme spécifique, appelé compilateur, permettant de traduire un programme écrit en ce langage de haut niveau en langage assembleur a été developpé.
- Un autre programme, appelé préprocesseur, fonctionne souvent en amont du compilateur et permet une certaine automatisation de la génération du code à compiler. Exemple : répétition de morceaux de codes, inclure des fichiers de façon intelligente les uns dans les autres, enlever les commentaires etc

Langage de programmation et notion de compilation d'un programme

En résumé, la compilation se décompose en 4 phases :

- Traitement par le préprocesseur.
- Compilation.
- Assemblage.
- Edition de liens

BIOS (Basic Input Output System) : Quésako ?

Problème de la poule et de l'oeuf

Pour communiquer avec les périphériques matériels, le système utilise les pilotes (petits programmes) qui se trouvent sur un périphérique (disque dur, CD-ROM etc). Il charge les pilotes à chaque démarrage.

Comment charger un pilote qui se trouve sur un périphérique qui est inaccessible sans son pilote ?

Réponse : C'est le rôle du BIOS qui va prendre en charge à bas niveau les communications avec les périphériques au démarrage.

BIOS : Où est-il stocké ?

- Mémoire ROM (Read Only Memory) : Partie non modifiable du BIOS.
- EEPROM : Partie modifiable du BIOS. Lorsque l'on modifie cette partie, on parle de "flashage" du BIOS (rare).
- CMOS : Paramètre du BIOS (s'efface si on enlève la pile de la carte mère).

BIOS : Rôles ?

- Initialisation des composants de la carte mère, du chipset et de certains périphériques.
- Identifie tous les périphériques internes et externes connectés.
- Initialise l'ordre des priorités des périphériques d'entrée (Disque Dur, CD-ROM, clé USB etc).
- Charge en mémoire et fait exécuter par le processeur le **premier secteur** de démarrage valide (présence du nombre magique sur les 2 derniers octets) présent sur un périphérique. Un secteur de démarrage valide est appelé master boot record (MBR). Il comprend 512 octets.

BIOS : Rôles ? (bis)

Remarques :

- Le BIOS est capable de lire sur les périphériques car il possède des micro-pilotes permettant de lire **un** secteur sur les périphériques. Il est également capable de communiquer avec le clavier et l'écran (affichage simplifié).
- Les étapes précédentes sont appelés "l'amorçage" ou Power-on self-test (POST).
- les anciens BIOS ne pouvaient gérer des disques que de 1024 cylindres, 256 têtes et 63 secteurs. Le système ECHS (extended cylinder/Head/Sector) et finalement le LBA ont été conçus pour contourner ce problème.

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader

BIOS :Quésako ?
BIOS : Où est-il stocké ?
BIOS : Rôles ?
BIOS : comment y accéder ?
BIOS : types de réglages ?
BIOS : futur ?

BIOS : Rôles ? (bis)

De nombreux BIOS :

Notons qu'il existe de nombreux BIOS dans un ordinateur, en particulier dans les cartes d'extension :

- BIOS de la carte garphique
- Firmware du graveur de CD
- BIOS de la carte réseau (gros serveurs)
- ...

Dans ce cours, lorsque l'on parle du BIOS, il s'agit du BIOS de la carte mère.

BIOS : comment y accéder ?

Pression d'une combinaison de touches après mise sous tension de l'ordinateur et pendant le contrôle de la mémoire (POST).
Exemples :

- Suppr F1
- Suppr
- Ctrl + Alt + Echap
- Ctrl + Alt + S
- F2
- Ctrl + Entrée
- Alt + Entrée
- Alt + F1
- Ctrl + Alt + F1
- Ctrl + Alt + Inser
- F2
- ...

Combinaison mentionnée dans le manuel de la carte mère.

BIOS : types de réglages ?

Il est possible de régler de nombreux paramètres qui seront sauvés dans le CMOS :

- Réglage des tension : CPU, Mémoire,...
- Réglage de la fréquence et des multiplicateurs : Overclocking !
- Date, heure,...
- Ordre de priorité des périphériques d'amorçage.
- Mot de passe du BIOS (+remarque sécurité)
- ...

BIOS : futur ?

- A terme le BIOS est amené à disparaître et est remplacé depuis 2010 par l'UEFI (Unified Extensible Firmware Interface). (Voir plus loin).
- Période de transition : l'UEFI est capable d'émuler un BIOS.

Master Boot Record (MBR)

- Le formatage bas niveau permet de donner une structure régulière au disque dur.
- BIOS permet d'exécuter le premier secteur du disque (512 octets).
- Le premier secteur d'un disque partitionné sous le format "MSDOS" porte le nom de "Master Boot Record" (MBR) et possède un certain format.

Master Boot Record (MBR)

Le premier secteur (512 octets) d'un disque dur partitionné sous le format "MSDOS" a le format suivant :

Structure du MBR		
Position (octets)	Longueur (octets)	Description
000-445	446	code
446-509	64	Table de partition
510-511	2	mot magique

Le BIOS supporte ce format et considère le premier périphérique (présent dans les priorités d'armoçage) dont les deux derniers octets sont le nombre magique (0xAA55 en hexadécimal).

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Master Boot Record (MBR)
Extended Boot Record (EBR)

Master Boot Record (MBR)

Remarques :

- Le BIOS peut aussi booter (charger le bootloader) sur une disquette (qui n'est pas partitionnée). Dans le cas, il considère le VBR (Volume Boot record) de celle-ci.
- Le BIOS peut souvent aussi booter sur un CD-ROM amorçable : voir l'extension "El Torito" de norme ISO9660.

Code du Master Boot Record (MBR) : bootloader

- Code (446 premiers octets du MBR)=logiciel permettant de lancer le "chargeur d'amorçage" ou "bootloader" (qui va lui même lancer un système d'exploitation).
- Le code du MBR de Windows contient aussi des messages d'erreur et une signature du disque.

Exemples de bootloader :

- NTLDR : Windows NT loader.
- winload.exe : Windows VISTA.
- GRUB (GRand Unified Bootloader), GRUB2.
- LILO (Linux loader).

Table de partition du Master Boot Record (MBR)

- Le partitionnement consiste à créer une ou plusieurs de zones de stockage indépendantes sur un disque dur. Ces zones sont appelées "partitions". Un disque dur "MSDOS" doit au moins posséder une partition.
- Un disque dur "MSDOS" ne peut posséder au maximum que 4 partitions.
- Les partitions sont soit de type "primaire" soit "Étendue".
- Il y a au maximum 1 partition "Étendue".

Table de partition du Master Boot Record (MBR)

Table de partitions MSDOS		
Position	Longueur (octets)	Contenu
446-461	16	Entrée pour la partition 1
462-477	16	Entrée pour la partition 2
478-493	16	Entrée pour la partition 3
494-509	16	Entrée pour la partition 4

Rem. : Une seule partition au maximum peut être déclarée "Étendue".

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Master Boot Record (MBR)
Extended Boot Record (EBR)

Table de partition du Master Boot Record (MBR)

Structure d'une entrée de partition		
Position relative	Long. (octets)	Contenu
0	1	Indicateur de Boot (0x80=active)
1-3	3	Début de la partition en C/H/S
4	1	Type de la partition
5-7	3	Fin de la partition en C/H/S
8-11	4	1er secteur de la partition en LBA
12-15	4	Taille de la partition (en secteurs)

Michaël Quisquater (Maître de Conférences, UVSQ) Introduction à Unix/Linux: Cours 1.

Table de partition du Master Boot Record (MBR)

Remarques :

- Indicateur de Boot : 0x80 si partition active (bootable car présence d'un bootloader dessus) et 0x00 sinon. Maximum 1 partition active sur un disque.
- Type de la partition :
  - 0x00 vide
  - 0x07 (NTFS) : Windows
  - 0x83 (ext2, ext3,ext4) : Linux
  - partition étendue : 0x05 (adressage CHS)
  - partition étendue : 0x0F (adressage LBA)

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Master Boot Record (MBR)
Extended Boot Record (EBR)

Table de partition du Master Boot Record (MBR)

Remarques (suite) :

- Les valeurs maximales en C/H/S sont 1024 cylindres, 255 têtes et 63 secteurs. Si le premier secteur de la partition est au-delà, le début de la partition prend la valeur (1023,254,63) car le cylindre et la tête commence à zéro.
- Le système LBA permet théoriquement (4 octets) de coder plus ou moins 2.200Go (2.2 To) mais parfois le BIOS limitent à 137Go car LBA codé sur seulement 28 bits.

Partition Étendue

- Une partition étendue, dont la taille et la position sont déclarées dans l'entrée de la partition du MBR, est découpée en partitions logiques.
- Chaque partition logique va être précédée d'un "Extended boot Record" (EBR) dont la structure est similaire au MBR.
- Les EBR's vont former une liste chaînée dont la longueur est théoriquement infinie (en pratique l'OS impose une valeur limite). Le premier EBR de cette liste est le premier secteur de la partition étendue.

Histoire des ordinateurs
Fonctionnement du disque dur et formatage bas niveau
Langage de programmation/compilation
BIOS
Partitionnement MSDOS
Notion de GUID et UUID
Bootloader
Master Boot Record (MBR)
Extended Boot Record (EBR)

Partition Étendue

Structure de l'EBR		
Position relative	Longueur (octets)	Description
000-445	446	inutilisé (sauf si bootloader)
446-509	64	Table de partitions EBR
510-511	2	mot magique

Table de partitions EBR		
Position	Longueur (octets)	Contenu
446-461	16	Entrée pour la partition logique
462-477	16	si présente, référence l'EBR suivant
478-509	32	Inutilisé

Rem. : Chaque EBR est toujours le 1er secteur de la partition.

Michaël Quisquater (Maître de Conférences, UVSQ) Introduction à Unix/Linux: Cours 1.

Notion de GUID et UUID

- Les disques sont souvent référencés par le système en fonction de leur endroit de connexion (physique). Si on change cette connexion ou si on rajoute un périphérique, cela risque de ne plus fonctionner.
- Les types de partition dans le MBR sont désignés par un code assez court (ex. 0x0f). De ce fait, si il n'y a pas concertation entre les différents acteurs de l'informatique, il est possible que deux codes identiques soient attribués à des choses différentes.

Notion de GUID et UUID (suite)

Pour pallier à ces difficultés, l'idée a été d'attribuer un nombre unique à chaque disque et type de partition. Ces nombres, appelés GUID (Globally Unique IDentifier) pour le GPT (autre standard que parition MSDOS) et UUID (Universally Unique IDentifier) pour UNIX, sont choisis indépendamment de l'emplacement et suffisamment grand de telle sorte qu'il n'y ait pas de collision possible, même sans concertation.

Bootloader : Quésako ?

- Pour qu'un système d'exploitation puisse fonctionner, il faut que le noyau (voir plus loin) soit chargé en mémoire et qu'il s'exécute $\Rightarrow$ c'est le but du bootloader !
- Autres fonctions du bootloader :
  - Choisir entre plusieurs noyaux à charger
  - Passer des paramètres au noyau chargé
  - Choisir entre plusieurs OS (si plusieurs sont disponibles)
- Le bootloader est dépendant de la plate-forme matérielle (ex. Sun ou PC).

Processus de démarrage de la machine

1. Lors de la mise sous tension de la machine, le BIOS réalise l'amorçage de la machine en effectuant le POST.
2. Ensuite, le BIOS fait exécuter le premier secteur (MBR). Les premiers octets ne sont pas suffisants pour charger et lancer un noyau, ils constituent le "stage1" du bootloader.
3. Ce stage 1 n'est pas capable de gérer un système de fichiers et va donc chercher une seconde partie du bootloader placé sur le disque via son adressage (CHS ou LBA).
4. C'est cette seconde partie qui chargera le noyau en fonction de la configuration d'un fichier.

Stockage du bootloader GRUB2

- GRUB2 est une réécriture complète de Grub "legacy" et le remplace.
- La version PC (pour les architecture x86 ou x86-64 (amd64)) est dénommé "grub-pc".
- Grub2 est modulaire :
  - L'image amorçable "boot.img" qui se trouve dans le MBR.
  - le module principal de démarrage "core.img" (qui est capable de lire des systèmes de fichiers) qui se trouve entre la table de partition et le début de la première partition (premier secteur de la seconde piste en C/H/S).
  - Des fichiers de configuration se trouvant dans le partition du système.

Configuration de Grub2

- La configuration de Grub2 se trouve dans le fichier (pour linux) : /boot/grub/grub.cfg
- Ce fichier n'est jamais à modifier directement.
- A la place, on va utiliser des scripts qui se trouvent dans répertoire :

  /etc/grub.d

  via les opérations :

  1. Modifier le fichier /etc/default/grub en mode root.
  2. Lancer la commande update-grub (ou mieux grub-mkconfig) en mode root. Cette commande lance des scripts présents dans /etc/grub.d

Configuration de Grub2

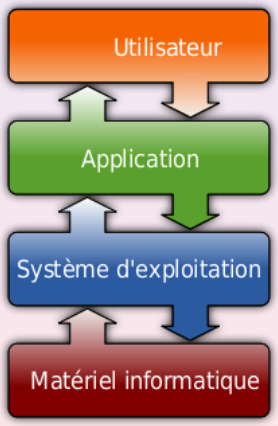
Fichier /etc/default/grub à modifier si nécessaire :

1. GRUB_DEFAULT=0 : Définit l'entrée sur laquelle on boote automatiquement.
2. GRUB_HIDDEN_TIMEOUT=10 : Nombre de secondes pour faire apparaître le menu. Si commenté alors affiché.
3. GRUB_HIDDEN_TIMEOUT_QUIET=true : Affichage compte à rebours.
4. GRUB_TIMEOUT=10 : Nombre de secondes à attendre avant de booter sur le default.
5. GRUB_DISTRIBUTOR='lsb_release -i -s 2>/dev/null || echo Debian' : distribution
6. GRUB_CMDLIN_LINUX_DEFAULT="quiet splash" options permanentes à transmettre au noyau lors du démarrage (acpi, verbosité,...). Ne concerne pas le mode "recovery".
7. GRUB_CMDLINE_LINUX="" options permanentes à transmettre au noyau lors du démarrage (acpi, verbosité,...). Concerne aussi le mode "recovery".
8. #GRUB_GFXMODE=640x480 : Spécifie la résolution du terminal graphique gfxgrub. Par défaut, cette résolution est trouvée automatiquement.
9. #GRUB_DISABLE_LINUX_UUID=true : n'attribue plus de UUID à la racine du système de fichier Linux.
10. #GRUB_DISABLE_RECOVERY="true" : enlève le mode recovery (c'est déconseillé !)
11. #GRUB_INIT_TUNE="480 440 1" : Fait du bruit quand GRUB démarre.

Deuxième partie II

Architecture des systèmes d'exploitation

Introduction



Le **système d'exploitation (OS : Operating System)** est chargé d'assurer la liaison entre les ressources matérielles (processeur, mémoire, périphériques), l'utilisateur et les applications. Lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques aux périphériques, il suffit d'envoyer les informations à l'OS qui s'en charge.

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix
Introduction
1ère génération : Batchs
2ème génération : Multi-tâches
3ème génération : Multi-utilisateurs ou en temps partagé
4ème génération : Temps réel
5ème génération : Distribué

Introduction

Il existe 5 générations de systèmes d'exploitation. Chacun des principes mis en oeuvre dans une génération se retrouve dans les générations suivantes.

Système d'exploitation batchs

- OS basés sur les batchs (lots) datent des années 50.
- Les programmes sont sauvegardés sur cartes perforées.
- OS basé sur les batchs est un programme moniteur, qui reste en permanence en mémoire centrale, et qui permet de commencer ou d'arrêter l'exécution du lot.
- L'opérateur place la pile de cartes dans le lecteur et puis demande d'exécuter le programme une fois celui-ci en mémoire centrale. A la fin de l'exécution, l'OS fait du nettoyage.
- Durant la lecture des cartes, le processeur central est au repos (donc 90% du temps).

Système d'exploitation multiprogrammés ou multi-tâches

- OS multiprogrammés datent des années 60 (circuits intégrés, stockage sur disque).
- Plusieurs programmes sont placés en mémoire centrale simultanément.
- Ordonnancement : Quand un programme attend des résultats provenant d'un périphérique, un autre programme est lancé $\Rightarrow$ usage du processeur optimisé. Le choix des programmes lancés se fait sur base de priorité et est réalisé par un **ordonnanceur (scheduler)**

Système d'exploitation multiprogrammés ou multi-tâches (suite)

- Mode d'exécution : Les programmes utilisateurs sont exécutés dans un **mode utilisateur** (**mode non-privilégié**), dans lequel l'exécution de certaines instructions est interdite. Les programmes tournant pour le compte du système sont exécutés en **mode noyau** (**mode privilégié**).
- Gestion mémoire : Protection de sorte qu'un programme n'écrive pas dans la mémoire attribuée à un autre.
- Technique du DMA (direct memory access) : le périph. transfert ses données en mémoire sans utiliser le processeur central $\Rightarrow$ usage du processeur optimisé.

Système d'exploitation multi-utilisateurs ou en temps partagé

- OS multi-utilisateurs datent des années 70.
- Notion d'utilisateur : Il s'agit de répondre rapidement aux utilisateurs de telle sorte qu'ils aient l'impression d'être les seuls sur le système.
- Ces systèmes mettent en oeuvre des techniques de multi-programmation.
- L'ordonnanceur est d'un type différent : le processeur va exécuter successivement des morceaux de programmes de telle sorte que l'on ait l'impression que ceux-ci sont traités simultanément.

Système d'exploitation multi-utilisateurs ou en temps partagé (suite)

- **SWAP** : lorsque le programme en cours d'exécution a besoin de plus de mémoire que ce qui lui est alloué alors un autre programme inactif est retiré et mis sur le disque dur. Ce transfert prend beaucoup de temps.
- Unix est un système en temps partagé.

Système d'exploitation temps réel

- OS en temps réel datent du milieu des années 70.
- Objectif : fournir des résultats corrects dans un temps imparti.
- Utilisation : pilote automatique, robots, applications vidéo et audio.
- Ces systèmes évitent la technique de SWAP qui est trop lente.
- Exemple : Windows CE, Embedded Linux, Symbian OS, PalmOS et VxWorks

Système d'exploitation distribué

- OS distribués datent des années 90.
- Il s'agit de gérer plusieurs ordinateurs à la fois.
- Ces OS permettent le partage des ressources entre les utilisateurs.
- Ces OS sont à l'origine destinés à de très gros systèmes. Aujourd'hui, il est de plus en plus courant qu'un ordinateur personnel dispose de plusieurs processeurs ou coeurs nécessitant une gestion particulière de la part de l'OS. Ces derniers systèmes ne sont pas à proprement parlé des systèmes distribués même si ils s'en rapprochent beaucoup.

Composants d'un OS

Noyau

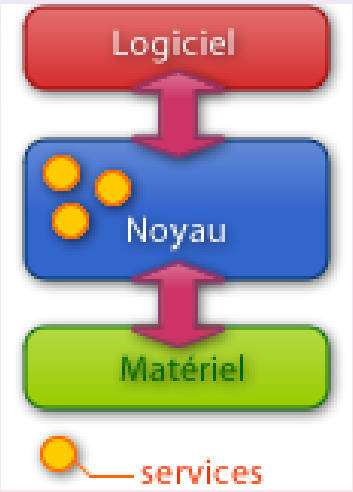
- Gestionnaire des ressources bas niveau.

Espace utilisateur

- Interface(s) utilisateur(s),
- Serveurs,
- Applications (logiciels, outils).

Remarque : La distinction entre les composantes de l'espace utilisateur n'est pas aussi claire. Il arrive souvent que certaines composantes soient à la jonction.

Rôle du noyau d'un OS



Le noyau d'un OS (Kernel en anglais) est le logiciel qui assure les fonctions de base concernant :

- La communication entre les logiciels et le matériel,
- La gestion des divers logiciels (tâches) d'une machine (lancement de programmes, ordonnancement, etc),
- La gestion du matériel (mémoire, processeur, périphérique, stockage,...).

Rôle du noyau d'un OS (suite)

En particulier :

- Gestion des processus : ordonnancement.
- Gestion de la mémoire.
- Gestion de systèmes de fichiers.
- Gestion des entrées/sorties,
- Gestion des supports réseaux (TCP/IP, etc),
- Gestion des droits,
- ...

Notion d'utilisateur

Dans les OS multi-utilisateurs, on distingue plusieurs types d'utilisateurs :

- Utilisateur Root (root user) : Il s'agit de l'utilisateur qui dispose de toutes les permissions sur le système.
- Utilisateurs usuels (normal user) : Il s'agit des utilisateurs qui utilisent le système.

Remarques :

- On appelle "administrateur" celui qui dispose du compte "root". C'est lui qui est en charge d'entretenir l'OS et de le faire évoluer.

Notion d'utilisateur(suite)

- Utilisateur de service (service user) : Il s'agit de comptes utilisateurs qui sont créés pour rendre des services (serveur web, serveur de courrier électronique etc). On ne peut pas accéder à ces comptes comme des comptes classiques.
- Pseudo-utilisateurs (pseudo-users) : Il s'agit par exemple des démons qui sont des applications qui tournent en tâches de fond. Ils ont tendance à disparaître au profit des utilisateurs de service.

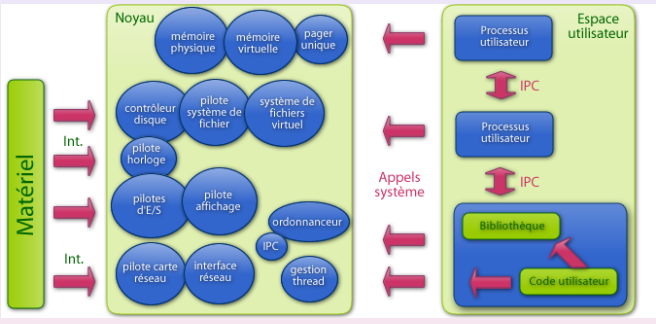
Remarque : Un utilisateur qui ne fait qu'utiliser le système (sans le modifier) ne touche pas à ces derniers utilisateurs.

Architecture des noyaux

Il existe des noyaux avec des architectures de différents types :

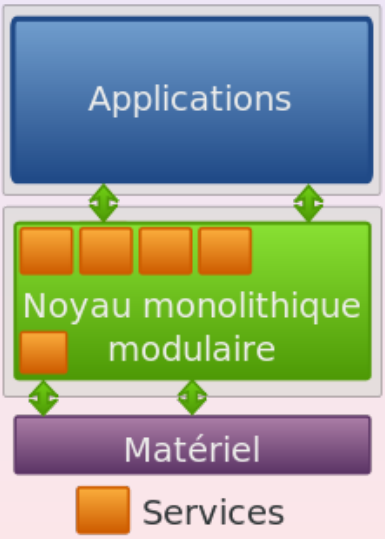
- Noyaux monolithiques non modulaires,
- Noyaux monolithiques modulaires,
- Micro-noyau,
- Noyaux hybrides,
- Noyau temps réel,
- ...

Noyaux monolithiques non modulaires



- Absolument toutes les composantes utiles au noyau sont dans le noyau.
- Si on souhaite rajouter une option (un pilote p.ex) au noyau, il faut le recompiler complètement.
- L'avantage de cette architecture est sa simplicité (un seul morceau).

Noyaux monolithiques modulaires



- On peut ajouter des modules au noyau.
- Ces modules seront chargés par le noyau en cas de besoin.
- L'avantage de cette architecture est sa flexibilité.
- **Exemple :** Noyau Linux depuis la version 2.0

Mode noyau et utilisateur

- La processeur peut fonctionner en deux modes, l'un dit "mode noyau", l'autre "mode utilisateur".
- Ces deux états sont matérialisés par un bit du registre d'état du processeur.
- En mode noyau, le processeur n'a pas de limitation au niveau des instructions, d'accès à la mémoire et des entrées sorties. En mode utilisateur, le processeur a des limitations.
- Sous Linux, toutes les applications s'effectuent en mode utilisateur (même l'utilisateur root) tandis que seul le code du noyau (et ses modules et pilotes) s'exécute en mode noyau.

Mode noyau et utilisateur (suite)

- **Mode noyau** $\Rightarrow$ **Mode utilisateur** : toujours possible car pas de limitation d'accès au matériel.
- **Mode utilisateur** $\Rightarrow$ **Mode noyau**. 3 cas possibles :
  1. Un processus utilisateur sous-traite au noyau la réalisation de certains tâches privilégiées (entrées-sorties, etc). Ces demandes se font via des fonctions **"appels systèmes"**. Celles-ci sont :
     - appelées depuis un programme de l'espace utilisateur,
     - exécutées dans l'espace noyau,
     - retournent le résultat dans le programme appelant (espace utilisateur).
  2. Un processeur détecte une erreur de fonctionnement due au processus en cours d'exécution,
  3. L'environnement extérieur (périphériques) doit indiquer au processeur l'occurence d'une situation spécifique (données disponibles etc)

Espace noyau et utilisateur

Espace noyau et utilisateur

Aux deux modes de fonctionnement correspondent deux visions de la mémoire par un programme en cours d'exécution (processus) :

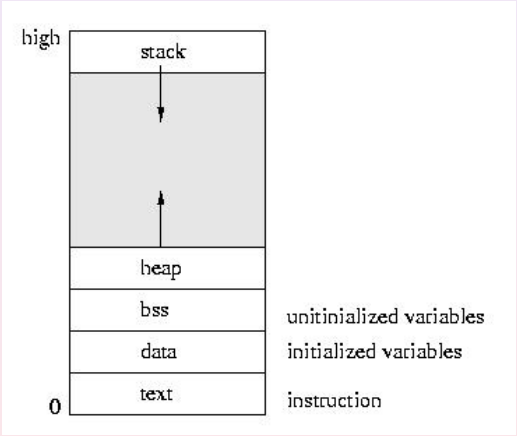
- Le noyau muni de ses modules et pilotes dispose d'un gestionnaire de mémoire qui est le seul à avoir accès à un espace noyau qui a les propriétés suivantes :
  - accession à l'ensemble de la mémoire disponible tant en lecture qu'écriture (même si celle-ci est utilisée),
  - accession à l'ensemble des propriétés de cette mémoire (vitesse, mode d'accès),
  - différenciation des zones de mémoire (pages, RAM, mémoire virtuelle).

Espace noyau et utilisateur (suite)

Mode user et noyau

- L'espace utilisateur est réservé aux processus correspondants aux applications (mode utilisateur). Le gestionnaire de mémoire donne l'illusion aux processus de manipuler une mémoire ayant les propriétés suivantes :
  - la mémoire allouée commence systématiquement à l'adresse 0,
  - la mémoire peut être indéfiniment étendue,
  - la mémoire est privée (protégée), un processus ne peut pas accéder à la mémoire d'un autre processus (sauf allocations et autorisations spécifiques).

Détail de l'espace utilisateur



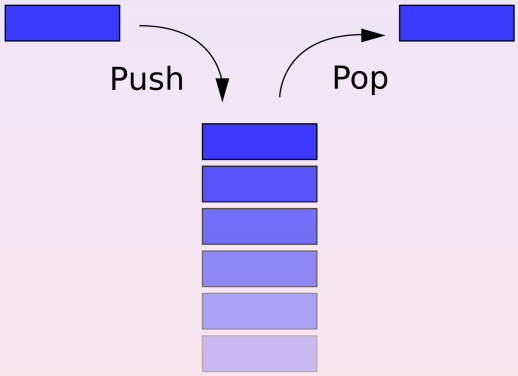
- Stack (Pile) : zone dans laquelle s'exécute le programme.
  - paramètres, variables locales, adresse, valeur retour de fonction
  - croît à chaque appel, décroît à chaque retour
- Heap (Tas) : zone où le programme alloue toutes les données qui ne sont pas en pile. ex : malloc.
- Text : zone où se trouve le programme à exécuter (souvent accès en lecture seule).

Détail de l'espace utilisateur

L'exécution d'un programme se déroule de la façon suivante :

- Le code comprenant les instructions à exécuter se trouve dans la région "Text". Ce code n'est pas instancié (écrit en terme de variables abstraites).
- Les variables globales (initialisées ou pas) se trouvent dans bss ou data.
- La mémoire allouée dynamiquement se trouve dans le tas (Heap)
- La pile (Stack) permet de gérer l'instantiation des fonctions (passage de paramètres), l'affectation des variables locales et les appels entre fonctions.

Pile d'exécution

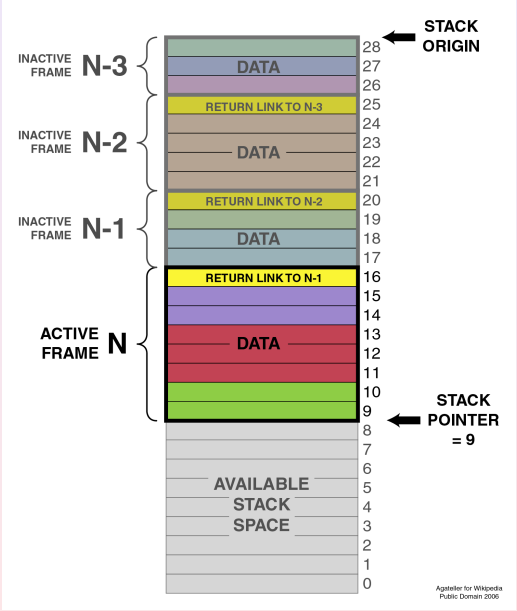


Il existe deux types d'opérations sur la pile :

- **Push** :
  consiste à ajouter un élément sur la pile
- **Pop** :
  consiste à extraire un élément de la pile

Remarque : Dans l'espace utilisateur, la pile se remplit par des adresses décroissantes (vers le bas dans le dessin précédent).

Pile d'exécution et appels de fonctions imbriqués



- L'appel de chaque fonction va être matérialisé par un "stack frame"
- Seul le dernier "stack frame" est actif et correspond à la fonction qui est effectivement exécutée,
- les "stack frames" inactifs correspondent aux fonctions appelantes imbriquées.

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix

Différents types de noyaux
Mode noyau et utilisateur/appels systèmes
Espace noyau et utilisateur
Pile d'exécution et appel de fonction
Noyau de l'OS en détail

Pile d'exécution et appels de fonctions imbriqués

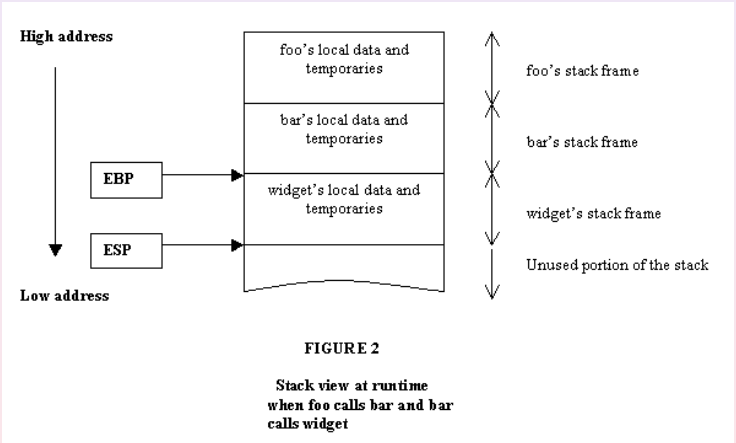


FIGURE 2

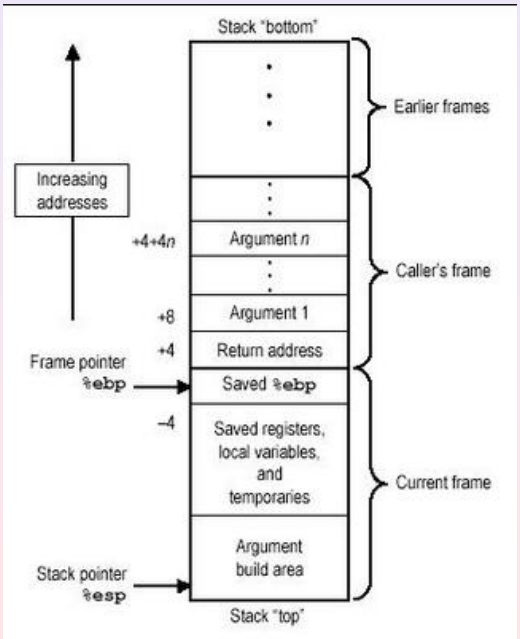
Stack view at runtime when foo calls bar and bar calls widget

- La base d'un "stack frame" est stocké dans le registre "EBP".
- Le haut de la pile (adresse minimale occupée) est stocké dans le registre "ESP".
- L'adresse de la prochaine adresse à exécuter (dans le "text") est stockée dans le registre "EIP".

Exécution d'une fonction

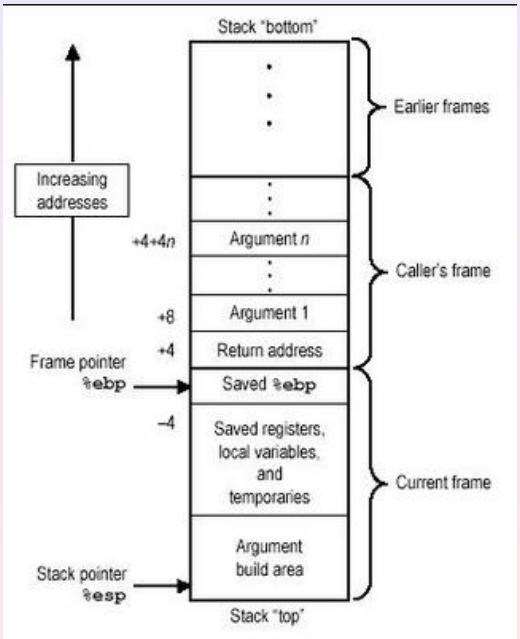
- Une fonction va prendre ses paramètres dans la pile.
- La fonction va utiliser la pile pour stocker ses variables locales.
- La pile va également servir à stocker certains registres si nécessaire.

Exécution d'une fonction en détail



1. Pusher les paramètres de la fonction dans l'ordre inverse (droite vers gauche), que l'on référencera grâce à l'EBP (voir étape 4.).
2. Appeler la fonction, ce qui va modifier "EIP" et pusher l'adresse de retour de fonction dans la pile (Return Address).
3. Pusher l'EBP courant (de la fonction appelante) dans la pile (c'est le Saved EBP).

Exécution d'une fonction en détail

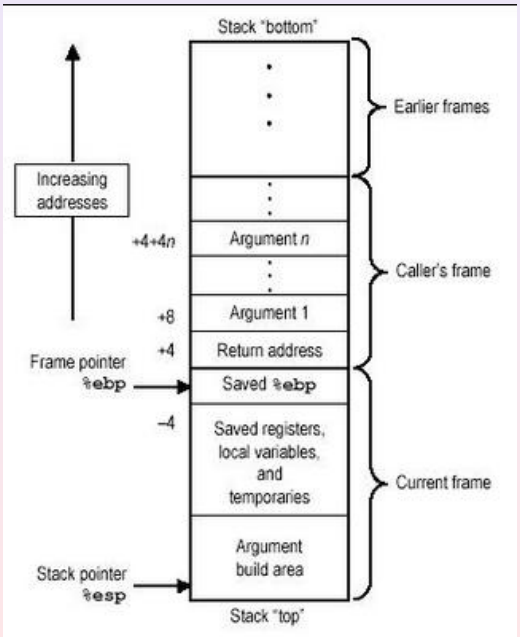


4. Reinitialiser l'EBP avec la valeur courante de l'ESP.
5. Stocker les variables locales, certains registres etc que l'on référencera grâce à l'EBP
6. Faire des calculs sur ces variables locales etc grâce au code de la fonction.
7. Renvoyer le résultat de la fonction dans le registre EAX (ou AX ou AL).

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix

Différents types de noyaux
Mode noyau et utilisateur/appels systèmes
Espace noyau et utilisateur
Pile d'exécution et appel de fonction
Noyau de l'OS en détail

Exécution d'une fonction en détail



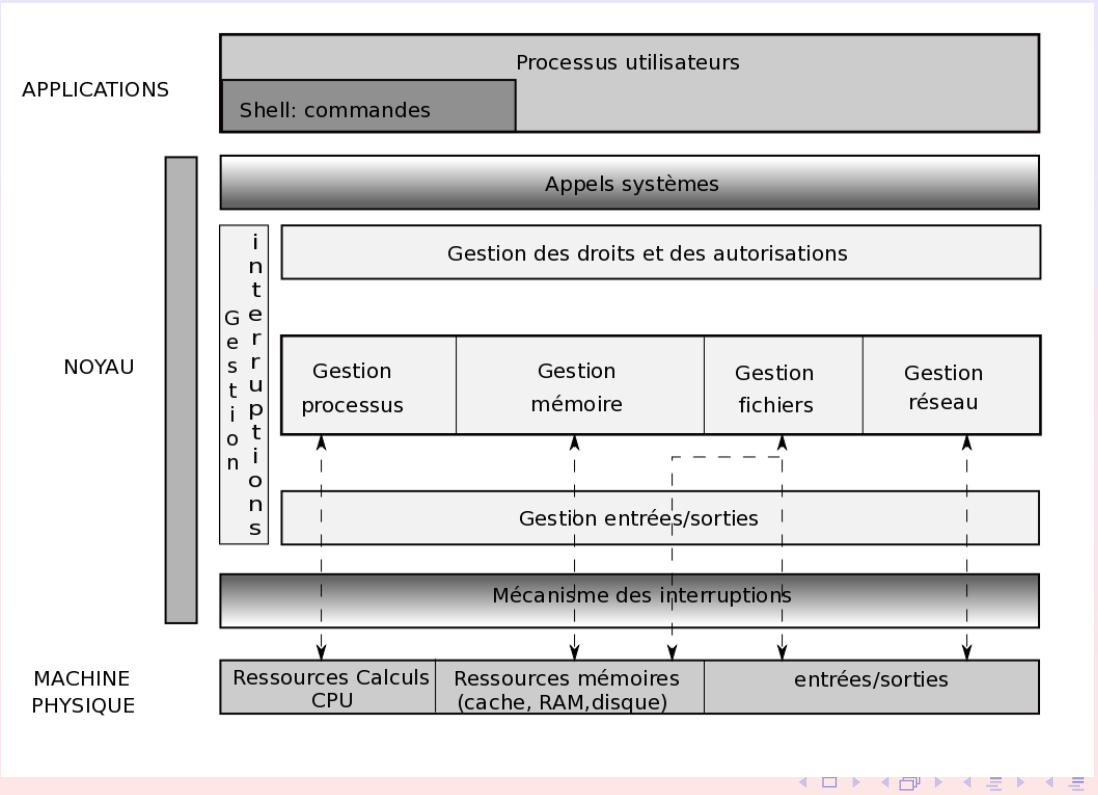
8. Reinitialiser les registres à partir des valeurs des registres de la pile.
9. Copier l'EBP courant dans l'ESP (cela consiste à dépiler la pile).
10. Mettre la valeur courante de la pile (Saved EBP) dans l'EBP courant en faisant un pop.
11. Mettre la valeur courante de la pile en faisant un pop dans EIP

Exécution d'une fonction en détail. Code source :

```
1 int f();
2
3 void g()
4 {
5  int x=5 , y;
6  y=f();
7  x=x+y;
8 }
```


Exécution d'une fonction en détail. Code assembleur :

```
push   %ebp              // Sauve l'ebp de l'appelant de g
mov    %esp, %ebp        // Mets à jour ebp
sub    $8,%esp           // Fais de la place pour x et y
mov    $5,-4(%ebp)       // initialise x
call   __Z1fv            // appelle la fonction f
mov    %eax,-8(%ebp)     // valeur de retour de f dans y
mov    -8(%ebp), %edx    // copie y dans edx
lea    -4(%ebp), %eax    // copie l'adresse de x dans eax
add    %edx,(%eax)       // ajoute y à x et mets dans eax
leave                    // quitte g c-à-d mov %ebp,%esp
                                 et pop %ebp
ret                      // pop l'adresse de retour et la mettre
                                   dans le compteur EIS
```



Gestion des processus

Un processus est un code informatique en cours d'exécution.

Un des composants du noyau va s'occuper de la gestion des processus. En particulier, il va s'occuper de :

- 'ordonnancement, qui consiste à choisir l'ordre d'exécution des processus sur le(s) processeur(s) d'un ordinateur. Il s'agit d'une allocation de la ressource "calcul". Il existe deux grandes classes d'ordonnanceurs :
  - temps partagé : Celui-ci consiste à exécuter les processus en "parallèle". Le processeur passe d'un processus à l'autre même si le processus en cours n'est pas terminé. Cela permet de donner l'impression de réaliser plusieurs tâches en même temps.
  - temps réel : Celui-ci fait en sorte que l'exécution d'un processus soit réalisé dans un temps imparti. Condition cruciale pour les systèmes embarqués.

Gestion des processus : ordonnancement (suite)

- l'envoi des signaux entre processus et entre le noyau et un processus
- Communication entre processus

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix

Différents types de noyaux
Mode noyau et utilisateur/appels systèmes
Espace noyau et utilisateur
Pile d'exécution et appel de fonction
Noyau de l'OS en détail

Gestion de la mémoire

Un des composants du noyau va s'occuper de la gestion de la mémoire. En particulier,

- Implémenter le modèle de la mémoire octoyé à un processus utilisateur.
- Faire en sorte que chaque processus n'interfère pas entre eux au niveau de la mémoire.
- Octroi de mémoire à un processus en fonction de certains paramètres.
- Permettre à des processus de communiquer.

Gestion de la mémoire

- Optimisation de la vitesse d'accession à la mémoire (mémoire RAM, cache, registre etc).
- Des échanges incessants entre ces différents niveaux de mémoire vont avoir lieu afin d'optimiser l'accès à la mémoire au niveau global. C'est le noyau qui va se charger des différents échanges.
- Ces échanges permettent aussi d'optimiser la ressource mémoire au niveau de la capacité. Si l'ordinateur ne dispose pas assez de mémoire pour l'ensemble des processus en cours, il va en stocker certains temporairement sur le disque. C'est le mécanisme du SWAP. Bien sûr, cela ralentit beaucoup l'exécution car le transfert sur le disque est lent.

Gestion des systèmes de fichiers

- Les données présentes sur un support sont souvent structurées sous la forme d'un système de fichiers.
- C'est le noyau qui va permettre de lire et d'écrire sous cette forme structurée.
- Le terme "système de fichiers virtuel" est utilisé lorsque l'information est rangée en mémoire vive à l'inverse du terme "système de fichiers" qui réfère généralement à la façon dont les données sont stockées sur un support permanent (disque, CD-ROM, clé usb,...).

Remarque : les systèmes de fichiers seront davantage détaillés au cours suivant.

Gestion des supports réseaux

- Les noyaux de nouvelle génération permettent de gérer le support réseau (TCP/IP)
- Certains réseaux permettent aussi des services élaborés tels que le partage de fichiers en réseau.

Gestion des interruptions

- logicielles
  - Appels systèmes
  - Opération illicites dans un processus (p.ex division par zéro)
- matérielles (les périphériques)

Gestion des droits et des autorisations

- Certains systèmes de fichiers permettant d'attribuer des droits d'accès au fichiers. C'est le noyau qui va vérifier si une requête (lecture/écriture/exécution) est autorisée ou pas (en fonction de l'utilisateur).

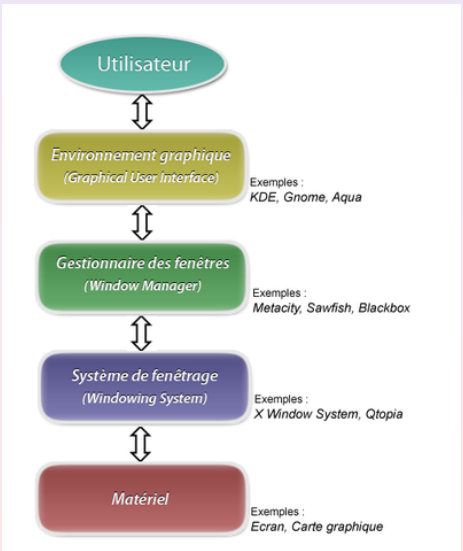
Gestion des entrées/sorties

- L'ordinateur ne peut pas naturellement communiquer avec un périphérique. Il ne connait pas le langage que le périphérique comprend.
- Le rôle du noyau est d'exécuter des codes informatiques, appelés **pilotes ou driver**. Ces codes établissent le lien entre le "langage" de la machine et celui du périphérique.
- Ces pilotes se chargent aussi de la mise en place de tampons mémoires.
- Certaines optimisations sont également à la charge de cette partie du noyau. Ex. : scheduler pour le disque afin de minimiser les mouvements du bras p.ex

Terminal/Console

- La console d'un PC est le moniteur de l'ordinateur.
- Les terminaux en mode texte présentent les sorties uniquement sous forme textuelle et disposent simplement d'un clavier pour les entrées.
- Un exemple de terminal texte qui fut répandu en France est le Minitel, lequel est relié aux serveurs par l'intermédiaire de la ligne téléphonique.
- Dans le monde informatique, le VT100 de DEC a été énormément utilisé, et est devenu un standard de fait.

Architecture système de fenêtrage/gestionnaire de fenêtres/environnement graphique



- **Environnement graphique/bureau** : Logiciels facilitant des tâches via une interface graphique (trouver une application, calculatrice, etc).
- **Gestionnaire des fenêtres** : Logiciel permettant de gérer différentes fenêtres graphiques (affichage, disposition, placement, actions, aspect graphiques (dock), ...).
- **Système de fenêtrage** : Affiche ce que le noyau (ou un autre système) lui demande. Gère le clavier et la souris.

Émulateur de terminal

Un émulateur de terminal, aussi appelé console virtuelle, est un logiciel qui émule le fonctionnement d'un terminal informatique.

Remarque : L'émulation est généralement faite dans une fenêtre et fonctionne donc au-dessus d'un système de fenêtrage ou d'un gestionnaire de fenêtres.

Outils

- L'utilisateur peut également disposer de logiciels comme :
  - Un compilateur,
  - (Des pilotes),
  - Un gestionnaire de fichiers,
  - Un gestionnaire de mise à jour,
  - Des outils de développement,
  - Une calculatrice,
  - Des éditeurs de texte,
  - ...
- Linux=Noyau, GNU/Linux=Système d'exploitation (noyau Linux et logiciels GNU).

La naissance de Unix

- 1964 : MIT, Bell Labs et General Electric : OS en temps partagé Multics (MULTiplexed Information and Computing Service). Centaines d'utilisateurs (zone Boston).
- 1969 : Bell Labs se retire du projet Multics. Ken Thompson (Bell Labs) développe un OS mono-utilisateur ; NEW Ken's System sur mini-ordinateur PDP-7.
- Souhait de Thompson de réécrire l'OS en FORTRAN mais conception du langage B avec D. Ritchie pour réécrire l'OS (jamais écrit en B en fait).
- Ritchie écrit le NB (New B) qui fut rebaptisé en C.
- Brian Kernighan suggère le nom UNICS (une seule chose d'une seule façon) à l'OS développé par Thompson et Ritchie en opposition à Multics. L'OS sera finalement baptisé UNIX.

La naissance de Unix (suite)

- Un décret interdisait AT&T (dont dépendait Bell Labs) de vendre autre chose que des équipements téléphoniques ou télégraphiques. ⇒ distribution en 1975 de UNIX dans les universités avec le code source moyennant une licence très faible.
- En 1975, Ken Thompson est invité à UCB (université de Californie Berkeley) pour un an.
- En 1977, création de la Berkely Software Distribution (BSD), fruit de nombreuses améliorations de la version de AT&T.

L'expansion et développement commercial

Trois branches de développement des sources virent le jour :

- Branche de recherche d'AT&T (Bell Labs) jusqu'en 1990.
- Branche commerciale d'AT&T qui développa le System III (1982) puis quatre éditions du SYSTEM V (1983).
- BSD développé par l'université de Californie jusqu'en 1994.

Variantes de UNIX développées par des entreprises :

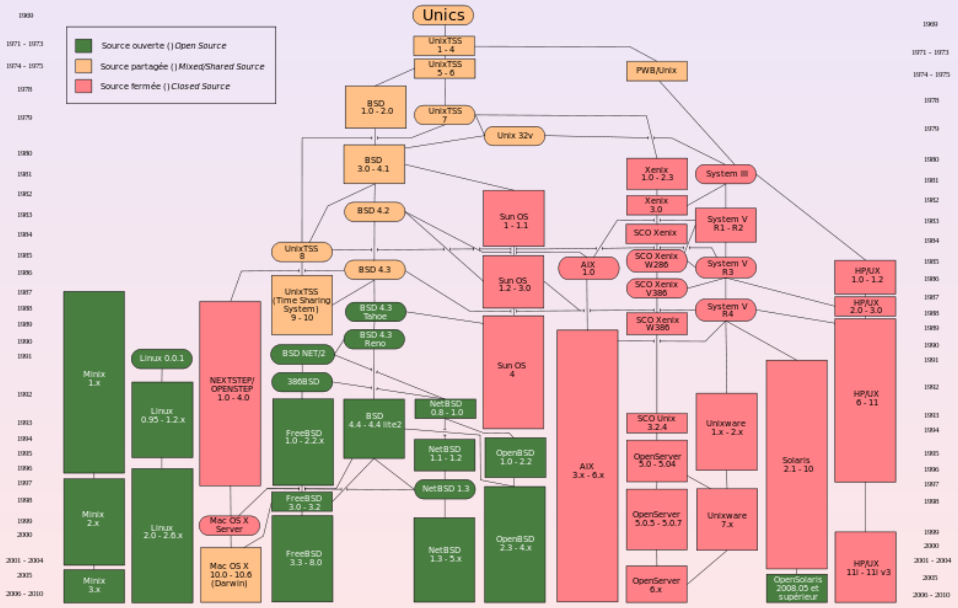
- XENIX par Microsoft (1980) : Base UNIX (AT&T recherche). Fonctionne sur les processeurs Intel 8086, Motorala M68000 et Zilog Z8000.
- Solaris par SUN (1981) : base BSD.
- A/UX, UNIX développé par Apple et compatible Mac OS.
- Amiga UNIX
- ...

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix
Histoire
Norme Posix

La naissance de Linux

- 1985, Tanenbaum développe un système d'exploitation minimal, Minix, afin d'enseigner les concepts des OS.
- 1991, conception par un étudiant finlandais, Linus Torvals, d'un OS fonctionnant sur les processeurs d'architecture Intel 80386 sur base du modèle Minix. "Hello everybody out there using minix - I'm doing (free) operating system (just, won't be big and professional like gnu) for 386(486) AT clones".
- Linux n'étant qu'un noyau, il utilise l'ensemble des logiciels (emacs, gcc, gdb, bash,...) du projet GNU (GNU is not Unix).
- 1992, réunion de programmeurs pour former l'équipe XFree86. Objectif : créer serveur X pour systèmes Unix tournant sur PC.
- En 1992-1993, plusieurs distributions sont proposées : Slackware, SuSE, Debian, Red Hat.

Généalogie de UNIX



Norme Posix et Unix

- Un OS effectue de nombreuses actions sur demande des logiciels applicatifs.
- Point de contact entre plusieurs logiciels applicatifs est appelé interface de programmation ("Application Programming Interface", API) ⇒ bibliothèques logicielles ou logiciels serveur répondant aux requêtes.
- L'utilisation de la même interface quel que soit le matériel, le protocole ou le système de fichier assure la portabilité des logiciels applicatifs.

Norme Posix et Unix (suite)

- POSIX (Portable Operating System Interface, 1988) est une norme relative à l'interface de programmation de l'OS.
- Ce standard a émergé d'un projet de standardisation des API de logiciels destinés à fonctionner sur des variantes de UNIX. Il est considéré comme à l'intersection des API System V et BSD.
- le X de POSIX est là pour rappeler l'origine UNIX de la norme. UNIX est évidemment conforme à POSIX.
- POSIX spécifie les différentes interfaces utilisateurs et logicielles (API). Aussi, POSIX propose une suite de tests appelée PCTS (POSIX Conformance Test Suite).

Système d'exploitation : quésako ?
Architecture d'un OS actuel
Architecture du noyau
Interface(s) utilisateur(s)
Outils
OS Unix
Histoire
Norme Posix

Norme Posix et Unix (suite)

- **Exemple de commandes utilisateurs**
  - cmp : compare deux fichiers.
  - echo : affiche l'argument à l'entrée standard.
  - more : affiche un fichier à l'entrée standard une page à fois.
- **Exemple d'interfaces logicielles**
  - open(fichier, mode) : ouvre un fichier pour lire et écrire.
  - stat(nom,&tampon) : lit et renvoie des informations sur un fichier.
  - close(fd) : ferme un fichier ouvert.

Michaël Quisquater (Maître de Conférences, UVSQ) Introduction à Unix/Linux: Cours 1.

Compatibilité des systèmes UNIX et POSIX

Pleine compatibilité à la norme POSIX :

- A/UX (Apple)
- BSD/OS
- Solaris
- OS X (Apple)
- HP-UX
- ...

Presque pleine compatibilité à la norme POSIX :

- FreeBSD
- GNU/Linux
- MINIX
- OpenBSD
- OpenSolaris
- ...

Troisième partie III

Notion de machine virtuelle

Machine virtuelle

- Il s'agit d'un logiciel s'exécutant sur un ordinateur permettant d'émuler la présence de ressources matérielles telles que la mémoire, le processeur, le disque dur,...
- Si cette émulation correspond à un ordinateur existant, il est alors possible de faire tourner un système d'exploitation prévu pour la machine émulée sur cet ordinateur disposant de cette machine virtuelle (logiciel !).

Machine virtuelle(suite)

- Virtualisation complète (processeur+périphériques). Exemple : Qemu. Le système d'exploitation installé sur la machine virtuelle s'exécutera naturellement plus lentement que s'il tournait immédiatement sur la machine.
- Virtualisation partielle (uniquement les périphériques). Exemple : VMware, Virtualbox. Les performances du système d'exploitation installé sur la machine virtuelle sont très proches de celles du système hôte.

Dualboot

Dualboot

Il s'agit de partionner le disque en deux et de mettre un OS sur chacune. Le bootloader permet de sélectionnner l'OS à lancer lors du démarrage de la machine.

Wubi

Wubi

Il s'agit de mettre un OS dans un fichier (faisant parti du système de fichiers de l'OS hôte) et de le lancer au démarrage de la machine via un bootloader. La différence principale avec le dualboot est le fait de devoir passer par le système de fichiers de l'OS hôte même si celui-ci ne tourne pas. Ce système n'est pas complètement stable (beaucoup de mauvaises expériences sur le web : perte du fichier boot.disk par exemple).

Wine

Wine

Il s'agit de faire tourner des logiciels Windows sur linux. Le principe consiste à traduire les appels systèmes d'un système vers l'autre. La compatibilité n'est pas parfaite.

Source des images

- Wikipedia