

Plan de l'introduction

1. Historique. modules de base : processeur, mémoire, unités d'échange, bus.
2. Rôle d'un système d'exploitation
3. Caractéristiques
4. Exemple : Unix,
5. Interface utilisateur
6. Comment comparer: mesure des performances.

Introduction

- 1 Historique, modules de base : processeur, mémoire, unités d'échange, bus.**
2. Rôle d'un système d'exploitation
 - garantir l'indépendance vis à vis du matériel,
 - optimiser l'utilisation des ressources,
 - proposer une interface utilisateur souple et puissante,
3. Caractéristiques
4. Exemple : Unix,
5. Interface utilisateur
6. Comment comparer: mesure des performances.

Introduction

Les machines dédiées à l'aide au calcul ont longtemps coexisté avec les automates. Ils ont évolué en parallèle, les machines à calcul étant toujours actionnées par un opérateur humain. L'idée de piloter de façon automatique une suite de calculs revient à Charles Babbage et à Ada Byron.

On cite le métier à tisser de Jacquard comme archétype d'automate : les actions à enchaîner par le métier à tisser sont inscrites sur une carte en bois que l'on donne à "lire" à la machine. La machine va traiter les informations contenues sur les carte et gérer **seule** les mouvements de ses engrenages.

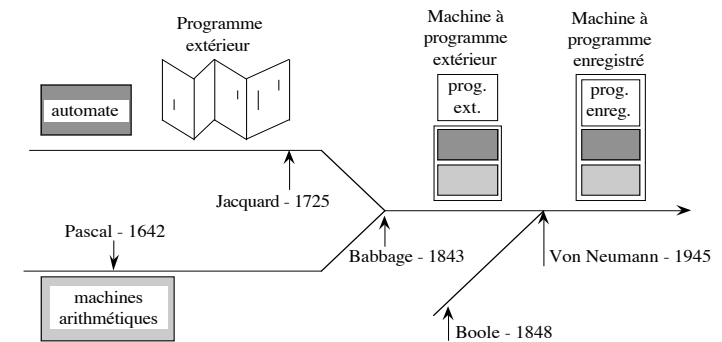
Notons deux points :

- il s'agit d'opérations de tissage, non pas d'opérations arithmétiques ou logiques.
- les actions ne sont pas mémorisées dans la machine, mais sur un support **externe** que l'on doit changer chaque fois que le travail à effectuer change, c'est-à-dire chaque fois que le programme change.

Introduction

Historique

- Automates et machines à calcul :



- La machine de Babbage est la première à mettre un automate au service du calcul :

Jusqu'aux années 30	Machines mécaniques
Années 30-40	Machines électromécaniques
Depuis les années 40	Machines électroniques

Introduction

Babbage et la comtesse Ada (qui a donné son nom à un langage informatique) ont conçu une machine dans laquelle un automate **pilote** une partie **calcul**.

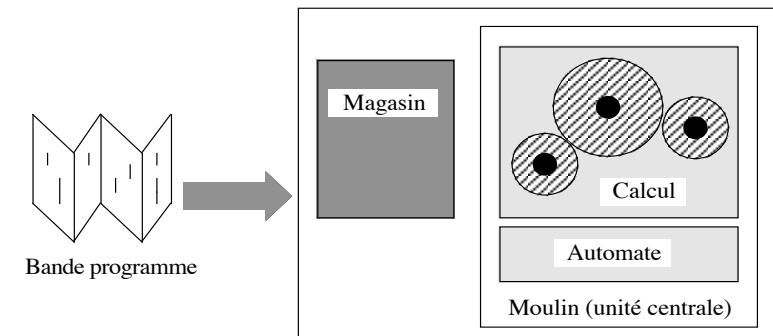
Cette machine ressemble fort à un ordinateur au point près suivant :

- le programme reste externe, c'est-à-dire que la machine ne mémorise pas les différentes suites d'opérations **arithmétiques** à effectuer. A chaque programme correspond une carte en bois qu'il faut donner à "lire" à la machine.

Introduction

La machine de Babbage

- Le premier automate au service du calcul, la machine à calcul de Babbage (et de la comtesse Ada) :



Magasin	stockage des résultats intermédiaires
Automate	pilotage de la partie calcul
Calcul	exécution des commandes

- Pas encore un ordinateur : le programme est extérieur.

Introduction

Remarque : notre exemple montre une machine de type Babbage traitant un programme C, bien sûr Babbage et Ada ne programmaient pas en C ! Le programme externe est écrit en C pour des facilités de lecture...

Le fonctionnement de l'automate de la machine de Babbage est le suivant :

- Lire une commande sur la carte. Cette commande est codée, c'est-à-dire représentée sous forme compréhensible par l'automate. Donc une commande telle que "ajouter x à y et ranger le résultat dans z" (écrite : $z = x + y$ en langage C) sera représentée sous forme de trous et d'absence de trou dans la carte en bois. (cf. le ruban perforé des orgues de Barbarie).
- Comprendre la commande (décodage),
- Charger les données si nécessaire. Par exemple, si la commande est : "ajouter x à y et ranger le résultat dans z", il faut aller chercher x et y,
- Sélectionner l'opération arithmétique à effectuer. C'est ici que l'on met x et y en entrée de la partie addition de la machine.
- Lancer le calcul : on donne l'ordre à la partie addition de faire la somme des informations présentées en entrée.
- Ranger le résultat du calcul dans le magasin (la mémoire des ordinateurs actuels). Le magasin sert à mémoriser les résultats des opérations. Ainsi, dans notre exemple, la sortie du module addition, rangée dans z, pourra être réutilisée pour une autre opération.

Cette machine n'est pas tout à fait un ordinateur, en particulier parce que les programmes ne sont pas rangés dans le magasin. La suite d'opérations à effectuer reste mémorisée sur un support **externe** à la machine. A chaque changement d'activité, il faut recharger manuellement un nouveau programme.

Cependant ce fonctionnement en cycles *fetch/decode/execute/store* (chercher une instruction/ la décoder/ l'exécuter/ ranger le résultat obtenu) est exactement celui des processeurs actuels.

Dans cette machine, les seules **informations** présentes en **mémoire** sont les **données**, elles sont représentées en **grisé** dans le schéma ci-contre.

Introduction

Fonctionnement de l'automate

- Lire une commande sur la carte.
- Comprendre la commande.
- Charger les données.
- Sélectionner l'opération arithmétique à effectuer dans « calcul ».
- Lancer le calcul.
- Ranger le résultat du calcul dans le « magasin ».

Introduction

L'équipe réunie autour de Von Neumann va construire une machine de conception nouvelle :

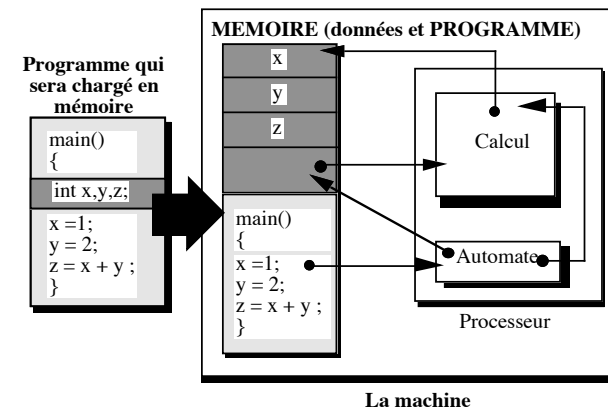
- les suites **d'instructions** agissant sur les données sont elles aussi **mémorisées** dans le "magasin",
- le magasin prend le nom de mémoire,
- toutes les informations seront codées en **binaire**, en particulier parce qu'il est difficile de distinguer plus de deux états stables en électronique,
- le jeu d'instructions traitées par la machine comprend au moins une instruction de branchement conditionnel (**instruction if** des langages de programmation).

En mémoire, rien ne permet de distinguer une donnée d'une instruction. Nous verrons que c'est le logiciel qui distingue des zones différentes en mémoire : région de code et de données et qui interdit d'écrire dans la zone de code ou d'exécuter des données !

Les deux types d'informations traitées par la machine sont les instructions et les données.

Introduction

Organisation de la machine : Von Neumann



- Modèle de Von Neumann :
- les programmes (suite d'instructions) et les données qui lui sont associées (entrées et résultats) sont rangés dans une mémoire unique et codés en binaire,
- Il existe au moins une instruction de branchement conditionnel.

Historique : les machines à calcul

- Les machines repères:

1641	Machine à additionner de Blaise Pascal
1806	Métier à tisser à cartons perforés de Jacquard
1837	Machine analytique de Charles Babbage
1928	Carte perforée à 80 colonnes et perforations d'IBM
1940	Le circuit imprimé
1942	Les diodes au germanium
1944	Mark 1, calculateur électromécanique (Harvard)
1946	ENIAC (Electronic Numerical Integrator And Computer), calculateur électronique (Pennsylvanie)
1947	EDVAC (Electronic Discret Variable Automatic Computer), calculateur à programme enregistré (Princeton)

Introduction

Rappel :

- dans une architecture de Von Neumann, les instructions et les données sont rangées dans la même mémoire.

La généalogie des systèmes d'exploitation reflète l'augmentation du parallélisme offert aux utilisateurs :

- simple enchaînement de tâches des traitements par lots,
- parallélisme logique de la multiprogrammation (UNIX),
- parallélisme vrai des systèmes gérant des machines multiprocesseurs ou des machines réparties sur un réseau.

Du point de vue de la diffusion et de la commercialisation des systèmes d'exploitation, on est passé de systèmes fonctionnant uniquement sur un type précis de matériel (systèmes dits propriétaires : MVS d'IBM, NT de Microsoft) aux systèmes portables (UNIX, Linux).

Le développement des architectures multiprocesseurs et des organisations multisites (ordinateurs interconnectés par un réseau) rend les problèmes de synchronisation, de communication et de partage des ressources de plus en plus complexes.

Il faut également noter l'intérêt de plus en plus grand porté aux systèmes embarqués (*embedded systems*) qui gèrent maintenant du matériel grand public : ordinateurs de bord sur les voitures, autocommutateurs, téléphones portables etc, alors qu'ils étaient récemment encore réservés au domaine militaire et à l'avionique.

Introduction

Historique : les dates

• Quelques repères :

1947	Le transistor
1950	La mémoire vive à tores de ferrite
1955	Le langage FORTRAN
1957	Le disque magnétique
1960	Le circuit intégré et le langage COBOL
1965	Le langage BASIC
1968	La mémoire à semi-conducteur
1970	Le disque souple
1972	Le microprocesseur
1973	Le circuit intégré à haut degré d'intégration LSI

• Les générations :

Première (1945-1955)	tubes électroniques.
Seconde (1955-1965)	transistors.
Troisième (1965-1980)	circuits intégrés.
Quatrième (1980-1990)	ordinateurs personnels et réseau

Introduction

La machine de Von Neumann modélise la plupart des ordinateurs actuels :

- la partie automate + calcul s'appelle **processeur** ou microprocesseur. La terminologie est variée, on parle également d'unité de traitement ou de séquenceur pour l'automate et d'unité de calcul ou d'unité arithmétique logique pour la partie calcul.
- la partie mémoire est la **mémoire RAM**,
- ces deux modules communiquent par l'intermédiaire de fils, l'ensemble de ces fils est appelé **bus**.

La mémoire **unique** contient toutes les informations dont a besoin le processeur :

- **données**,
- **instructions**,

Toutes ces informations sont codées en **binaire**. Rien ne permet de distinguer en mémoire une donnée d'une instruction.

Le processeur contient lui aussi des éléments de mémorisation : les **registres**.

Ces éléments jouent un rôle capital, citons :

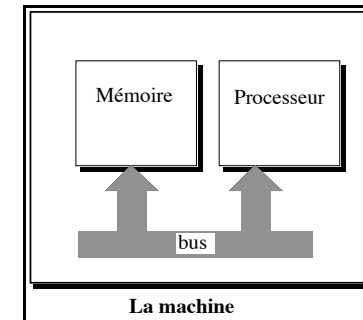
- le compteur ordinal (*program counter*) qui mémorise quelle instruction il va falloir effectuer
- le registre de pile (*stack pointer*) qui indique le début de la zone mémoire réservée à la pile

Il existe des machines dans lesquelles les données et les instructions sont rangées dans des mémoires distinctes : une mémoire pour les instructions, une pour les données, on parle alors d'architecture de **Harvard**.

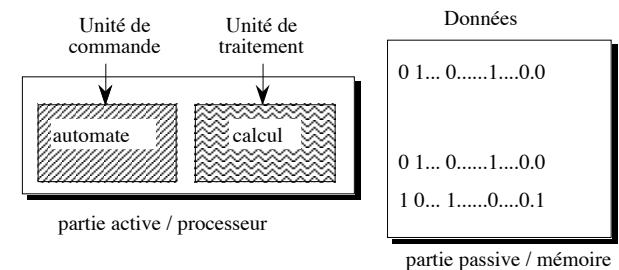
Introduction

Modules de base : processeur et mémoire

- Le processeur et la mémoire sont reliés par un ensemble de fils appelé **bus** :



- Toutes les informations sont représentées en **BINAIRE** :



Introduction

Le modèle de la machine ordinateur que nous avons construit ne correspond pas du tout à la perception qu'en a l'utilisateur : pour lui, la machine est avant tout un ensemble écran-clavier-souris attaché à une boîte dont le contenu et le fonctionnement sont totalement inconnus. C'est la partie principale du contenu de cette boîte (processeur et mémoire) que nous venons de décrire.

En fait l'ordinateur se présente à l'utilisateur par l'intermédiaire de ses moyens de communication, appelés périphériques. Ces périphériques sont variés : écran, clavier, disquette, souris, etc. Tous ces périphériques doivent être reliés au processeur par le bus, seul moyen de communication du processeur avec l'extérieur. Mais les périphériques n'ont pas les mêmes caractéristiques électriques, ni le même débit que le bus ; pour chaque périphérique il faudra donc un intermédiaire qui adapte son comportement à celui du bus : c'est ce qu'on appelle un contrôleur ou une carte contrôleur ; nous utiliserons la terminologie suivante :

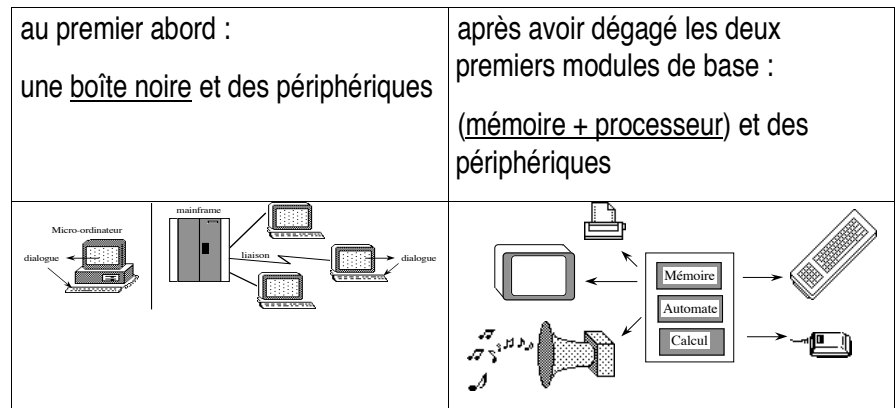
- **unité d'échange**

elle est le troisième module de base de notre modèle.

Introduction

Le point de vue de l'utilisateur

- Au premier abord, l'ordinateur apparaît comme :
 - Un ensemble d'outils de dialogue (clavier, écran, souris), reliés à :
 - Une boîte noire intelligente qui contient la mémoire et un organe de calcul.



- Que manque-t-il à notre modèle : les unités d'échange !

Introduction

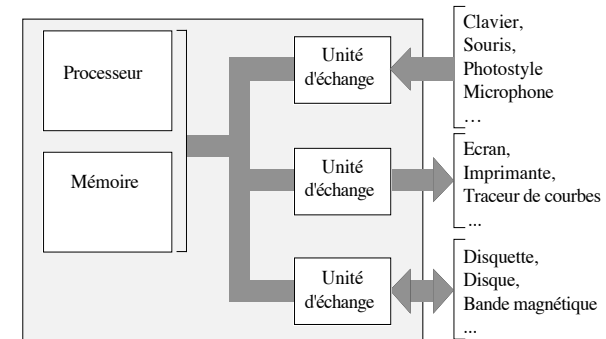
Les trois modules de base à partir desquels est construite la machine ordinateur sont :

- le **processeur** (un seul sur les machines monoprocesseur),
- la **mémoire**,
- l'**unité d'échange**, ou U. E , il y en a généralement une par type de périphérique ; mais pas nécessairement : cartes réseaux, cartes vidéo multiples, par exemple.

Introduction

Les modules de base : processeur, mémoire et unités d'échange

- Pour communiquer avec l'extérieur, on ajoute à la partie processeur/mémoire des moyens de communication reliés au bus par des unités d'échange :



- Rôle des UE : adaptation logique (débit, granularité des informations) et adaptation électrique.

Introduction

Les périphériques assurent :

- la permanence de l'information (son stockage),
- l'échange de l'information, c'est-à-dire la communication avec l'extérieur.

Insistons sur le fait que la mémoire RAM s'efface chaque fois que l'on coupe l'alimentation électrique, et que les disques sont des supports de mémorisation **permanents** (comme des bandes magnétiques), indispensables pour assurer la conservation des informations.

Les moyens de communications permettent le dialogue avec :

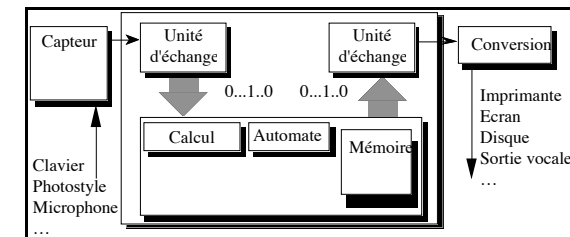
- d'autres ordinateurs (réseau),
- le monde extérieur (capteurs et émetteurs de tous types : ultra-son, radar, détecteur de choc, thermomètre, ...),
- un opérateur humain (clavier, écran, souris, microphone, haut-parleur, etc).

La mise en oeuvre des périphériques demande une bonne compréhension du fonctionnement du matériel.

Introduction

Les périphériques

- Sans moyens de communication avec l'extérieur la machine ne peut rien faire. On lui adjoint de nombreux PERIPHERIQUES, on parle d'unités d'entrées-sorties (*input-output*) :



- le clavier et l'écran sont deux périphériques différents,
- l'accès à un réseau est devenu un périphérique banal,
- importance des capteurs et émetteurs sur les systèmes embarqués,

Introduction

On distingue trois types de lignes sur le bus :

- **lignes de commandes.** Utilisées par le processeur pour indiquer à la mémoire ou aux UE s'il veut acquérir une information (lecture mémoire, acquisition souris ou clavier, accès disque en lecture, etc) ou s'il veut présenter une information (visualisation sur écran, accès disque en écriture, émission sur haut-parleur, etc),
- **lignes d'adresses.** Le processeur y indique le numéro de la case mémoire à laquelle il veut faire accès. Cet accès pourra se faire en lecture ou écriture.
- **lignes de données.** Ces lignes servent à faire transiter les informations dans le sens :
processeur -> mémoire ou UE
ou dans le sens mémoire ou UE -> processeur

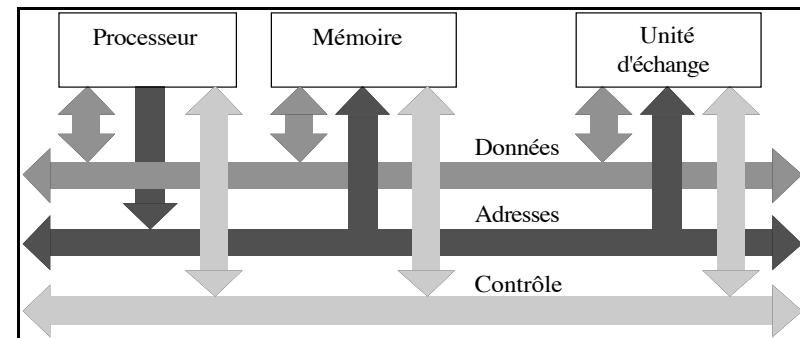
Exemple d'accès mémoire par le processeur :

- positionner l'adresse de l'information à traiter sur les lignes d'adresse,
- pour une lecture :
 - o indiquer « lire » sur les lignes de commande,
 - o attendre (**latence**) : le processeur est plus rapide que le mémoire,
 - o lire les données qui ont été présentées sur les lignes de données du bus par le contrôleur mémoire,
- pour une écriture:
 - o présenter les données à écrire en mémoire sur les lignes de données,
 - o indiquer « écrire » sur les lignes de commande,
 - o le contrôleur mémoire copie les lignes de données en mémoire.

Introduction

Les bus

- Les différents types de lignes sur un bus :



- **Comment se fait une lecture en mémoire :**

- positionner l'adresse de la cellule sur les lignes d'adresse,
- indiquer « lecture » sur les lignes de commande,
- attendre (**latence**) ,
- lire les données présentes sur les lignes de données,

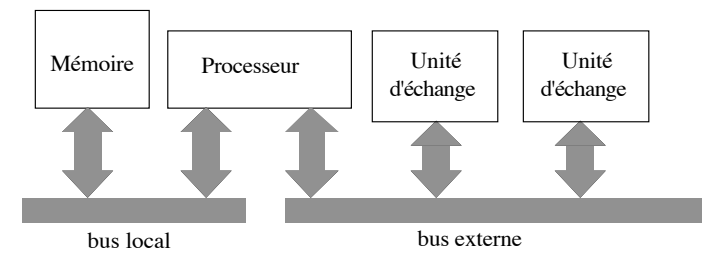
Introduction

Ce qui différencie les différents types de machines (ordinateurs personnels, stations de travail, serveurs de disques) ce sont plus les relations entre les différents modules que la puissance pure de leurs processeurs.

Introduction

Les bus

- les bus peuvent être nombreux :



Introduction

1 Historique, modules de base : processeur, mémoire, unités d'échange, bus.

+ . Rôle d'un système d'exploitation

- **garantir l'indépendance vis-à-vis du matériel,**
- **optimiser l'utilisation des ressources (en fonction des critères : équité, satisfaire les tâches les plus prioritaires,...)**
- **proposer une interface utilisateur souple et puissante**

3. Caractéristiques

4. Exemple : Unix,

5. Interface utilisateur

6. Comment comparer: mesure des performances.

Introduction

Une des définitions du mot système : combinaison d'éléments pour atteindre un objectif.

Nous verrons que les systèmes d'exploitation des ordinateurs (S.E, par la suite) n'ont pas tous le même objectif, on distinguera donc différents types de SE, citons les systèmes temps partagé (les plus courants) et les systèmes temps réel (souvent embarqués à bord d'une voiture, d'un téléphone portable, d'un avion, etc).

Rôle du système d'exploitation :

- **gérer au mieux un ensemble de ressources en fonction des services attendus.**

Le système d'exploitation rend accessible à l'utilisateur toutes les potentialités matérielles et logicielles de la machine. Intermédiaire entre le logiciel d'application et le matériel, il intervient ainsi comme outil de mise en œuvre et de **valorisation** du matériel : équipée de systèmes différents, la même machine peut apparaître plus ou moins attractive aux utilisateurs (puissance, fiabilité,...), c'est-à-dire qu'elle répond plus ou moins bien aux services attendus.

Le système consomme lui-même des ressources : espace mémoire, surface disque. L'espace mémoire dédié au système, ou espace résident, atteint couramment plusieurs Mo, la surface disque se chiffre en Go. Par exemple, pour faire agréablement fonctionner Unix sur un PC il faut au moins 128 Mo de RAM et 500Mo sur disque.

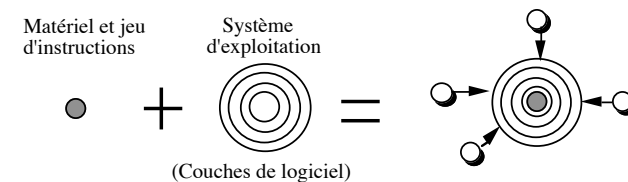
En ajoutant un S.E sur la machine physique, on construit une machine **virtuelle** (quel que soit le matériel, le type du processeur, l'utilisation de l'ordinateur se fait de la même façon). Parmi les fonctions de cette machine virtuelle, citons :

- la gestion du réseau qui permet à l'utilisateur d'accéder à un des machines distantes en donnant leur **nom**, et non pas en spécifiant leur adresse réseau.
- la gestion des fichiers qui permet à l'utilisateur d'accéder à un fichier en donnant son **nom**, plutôt qu'en donnant son emplacement sur le disque.
- l'écriture sur un périphérique qui se fera toujours via une primitive du type `write`, quelle que soit la structure physique de ce périphérique.

Introduction

Rôle d'un système d'exploitation

- assurer l'indépendance vis-à-vis des aspects matériels, c'est-à-dire construire une **machine virtuelle** sur la machine physique.



- organiser et optimiser l'utilisation des ressources (matérielles et logicielles) :
 - processeurs
 - mémoire
 - fichiers
 - communications
 - ...
- procurer une interface utilisateur conviviale.

Introduction

Ressource : quantité **limitée** d'un matériel ou d'un service qui doit être partagée.

Le nombre d'éléments d'une ressource est également limité : comment optimiser leur utilisation ? Le système est chargé de résoudre le problème de l'**allocation des ressources** : plus précisément il doit mettre en œuvre les ressources nécessaires à l'exécution d'un programme (surface disque, accès réseau, temps cpu).

On retrouve dans la gestion des ressources les composants matériels de base :

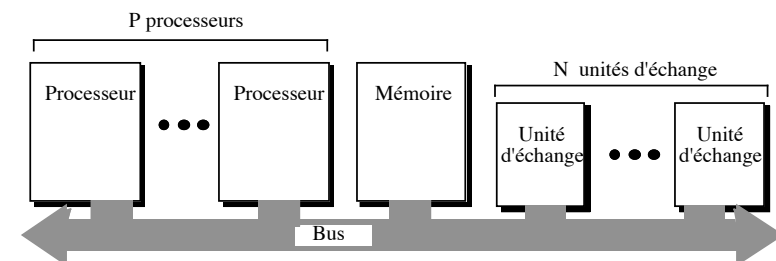
- gestion processeur(s),
- gestion mémoire,
- gestion fichiers,
- gestion périphériques divers.

L'utilisateur dispose de fonctions (primitives ou appels système) pour accéder aux ressources. Il connaît leur nom et la manière de les utiliser, mais il n'a pas besoin de connaître leur fonctionnement : il dispose d'une machine virtuelle.

Introduction

Les ressources

- Le système d'exploitation gère les ressources de la machine,
- Ressource : élément logiciel ou matériel fondamental :
 - la mémoire ,
 - le ou les processeur(s),
 - les périphériques,
 - les flux d'informations (les échanges entre la mémoire, les processeurs et les unités d'échanges),
 - la date et l'heure (estampillage),
 - les fichiers.



Introduction

1. Historique. modules de base : processeur, mémoire, unités d'échange, bus.

2. Rôle d'un système d'exploitation

- + **Caractéristiques**
 - services rendus
 - objectifs, modes d'exploitation
 - organisation

4. Exemples : Unix,

5. Interface utilisateur

6. Comment comparer: mesure des performances.

Introduction

Multitâche et multiutilisateur sont des caractéristiques logicielles, elles indiquent la capacité du système à gérer plusieurs travaux, ou plusieurs utilisateurs à la fois. Le fonctionnement multitâche est une condition nécessaire du fonctionnement multiutilisateur.

Multiprocesseur et multimachines (ou multisites) sont des caractéristiques matérielles, la première indique que le constructeur a équipé la machine de plusieurs processeurs, la seconde que l'utilisateur, ou que l'administrateur d'un parc de machines, a connecté une machine à un réseau d'ordinateurs.

La fiabilité traite la résistance aux pannes, la sécurité la résistance aux attaques.

Introduction

Caractéristiques d'un système

- Degré de parallélisme :
 - mono ou multi tâche,
 - mono ou multi utilisateur,
 - mono ou multi processeur,
 - mono ou multi machine.

- Services rendus :
 - facilité d'utilisation (interface utilisateur conviviale),
 - fiabilité, sécurité, déterminisme.

- Comment comparer ?
 - les MIPS et MFlops : indications sur le processeur,
 - utiliser des *benchmarks* (rôle du compilateur, débit d'entrées-sorties, ...)
 - transactions par seconde,
 - tests spécifiques : context switch... en temps réel.

Introduction

Dans un système temps partagé, c'est le S.E qui impose sa base de temps aux utilisateurs. L'interactivité avec la machine est conduite par celle-ci : on utilise les ressources (disque, clavier, écran) quand le **S.E le décide**.

L'objectif du système est le dialogue avec **tous** les utilisateurs, si ceux-ci sont nombreux l'allocation des ressources doit se faire de façon équitable.

Dans un système temps réel, c'est l'environnement qui **impose sa base de temps au système** : il doit donner le contrôle à la tâche qui est vitale à un instant donné. Sa contrainte principale sera de délivrer les résultats attendus avant une date limite appelée échéance, le non-respect de l'échéance pouvant mettre en danger l'installation contrôlée par la machine.

Soit un ordinateur qui gère les tâches suivantes sur un avion : projeter le film, faire le café, assister la descente. Si le pilote amorce la descente pendant la préparation du café, il faudra suspendre immédiatement ce travail pour passer à la gestion de la descente. Si la descente est mal gérée, c'est-à-dire si le pilote ne reçoit pas les informations d'altitude, etc, en **temps utile**, aucune autre tâche ne pourra plus s'exécuter...

Les objectifs imposés au système déterminent son orientation principale ; un système orienté temps partagé montrera sans doute de piètres performances en temps réel.

La destination d'une machine permet souvent d'orienter le choix de son système d'exploitation : MS-DOS convient à un PC destiné à des applications purement bureautiques (utilisation de Word, Excel, ...). Si ce PC sert de base de développement dans un environnement scientifique, UNIX, Linux seront sans doute une meilleure réponse. Pour piloter le système de freinage d'une voiture il faudra chercher une troisième voie...

Introduction

Quels objectifs ?

- Maximiser le nombre d'utilisateurs :

Système temps partagé

- mode le plus courant : utilisation interactive de la machine par un ou plusieurs utilisateurs

- Respect des échéances, interaction avec le monde réel :

Système temps réel :

- commande de processus industriel
- guidage d'engins
- autocommutateur téléphonique
- ...

- Cohérence des données, synchronisation des accès :

Système transactionnel :

- réservation de places de train, d'avion
- messagerie
- ...

Introduction

Le mode d'exploitation est imposé par les objectifs donnés au système. Plusieurs modes d'exploitation peuvent coexister sur une machine : la coexistence du traitement par lots (*batch*) et du temps partagé est l'exemple le plus fréquent. Certains systèmes orientés temps partagé offrent des possibilités pour faire fonctionner des applications de type temps réel.

L'objectif du temps partagé est la répartition équitable de ressources limitées entre des utilisateurs nombreux. On rappelle que le système consomme lui-même des ressources. Une trop grande charge en utilisateurs peut conduire à l'effondrement du système (thrashing).

Introduction

Modes d'exploitation

- Traitement par lots (*batch processing*) :
 - ensemble de tâches successives,
 - pas d'interaction entre le système et l'utilisateur,
 - résultats à la fin des travaux.

- Temps partagé (*Time Sharing, TS*) :
 - plusieurs utilisateurs simultanés,
 - chaque utilisateur dialogue avec *sa machine*,
 - notion de *quantum de temps*,
 - notion de *gestion* et de *partage des ressources*

- Temps réel (*Real Time, RT*) :
 - respecter des échéances (notion de *tâche critique*),
 - interagir avec l'extérieur,
 - pré-allouer les ressources,
 - fiabilité, déterminisme.

Introduction

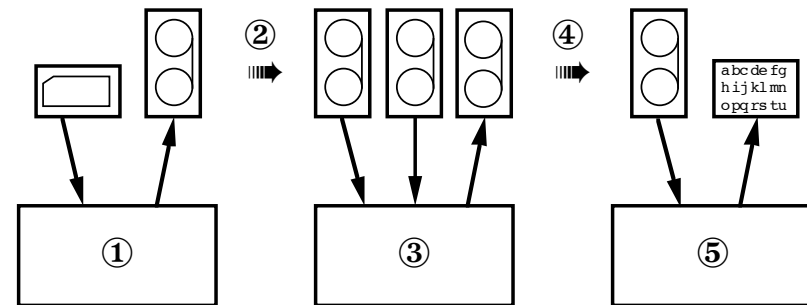
Nous sommes ici au temps des dinosaures de l'informatique : les systèmes d'exploitation des années soixante.

Le programme et les données sont enregistrés sur des cartes perforées. Le *JCL (Job Control Language)* permet d'enchaîner et de contrôler les différentes étapes des travaux à effectuer.

Des traitements simultanés se font sur des machines **différentes**. Chaque machine ne gère qu'une tâche à la fois, il n'y a pas déroulement de plusieurs activités simultanées sur une même machine.

Introduction

Les premiers systèmes d'exploitation : le traitement par lots



- 1 Les différents programmes à exécuter sont mis bout à bout puis transférés sur une bande magnétique. Cette opération est réalisée par un ordinateur spécialisé.
- 2 La bande magnétique est ensuite transportée sur la machine qui exécute les traitements. Le système permettant de faire fonctionner cette machine se trouve également sur bande magnétique.
- 3 Les résultats produits par l'exécution du « train de programmes » sont enregistrés sur une troisième bande magnétique.
- 4 La bande magnétique contenant les résultats est transportée sur un ordinateur spécialisé.
- 5 Le contenu de cette bande magnétique est imprimé.

Introduction

Le partage des ressources entre tâches concurrentes se fait sous les contraintes suivantes :

- 1- les offrir dans un délai raisonnable (éviter la famine),
- 2- optimiser l'utilisation de ces ressources,
- 3- exclusion mutuelle sur les ressources non partageables,
- 4- prévenir les blocages

L'exploitation temps partagé privilégie le point 1, ce qui peut être contradictoire avec 2. Le système se dote d'une politique d'allocation des ressources : pour chaque type de ressource un algorithme s'efforce de résoudre les conflits d'accès.

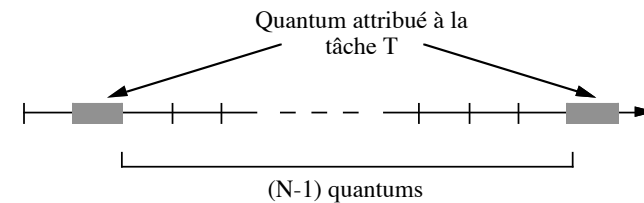
Le processeur est alloué aux travaux par unités de temps appelées quantum. Si il y a N travaux en cours et si un quantum vaut q millisecondes, chaque travail se verra attribuer un quantum tout les $(N-1)q$ millisecondes. Le temps de réponse reste agréable tant que N n'est pas trop grand !

Chaque utilisateur dispose de sa propre machine virtuelle qu'il personnalise en définissant un environnement de travail (fichiers de configuration, *look and feel* des systèmes de fenêtrage).

Introduction

Modes d'exploitation **Exemple 2 : le temps partagé**

- A chaque reprise de son exécution, une tâche dispose d'un quantum de temps q :



- N utilisateurs \rightarrow N machines virtuelles,
- Critère : temps de réponse "court",
- Gestion des ressources sophistiquée : mémoire virtuelle, accès au processeur en fonction de priorités variables, optimisation des accès disques, etc.
- Rôle de plus en plus important des interfaces : multifenêtrage, graphique,...

Introduction

On appelle logiciel de base l'ensemble des programmes qui proposent les fonctions nécessaires à l'ensemble des applications. On y trouve les compilateurs, les éditeurs, les langages de commandes, ... et bien sûr, le système d'exploitation. Quelles sont les fonctions du logiciel de base qui appartiennent au système d'exploitation ? Celles qui définissent et réalisent une machine virtuelle et celles qui assurent la gestion et le partage des ressources.

La frontière entre logiciel de base et logiciel d'application, ainsi que celle qui différencie les programmes du logiciel de base n'est pas toujours précise. Elle peut varier en fonction de la destination du système, de la fréquence d'utilisation des outils proposés par les différentes couches. Un utilisateur peut passer du logiciel d'application au logiciel de base ou vice-versa.

Le système est lui-même décomposé en plusieurs couches, en particulier pour des raisons de sécurité. UNIX est organisé en deux couches. La couche la plus proche du matériel s'appelle le noyau (*kernel*).

Le noyau, cœur du système, réside en mémoire, offre les mécanismes de base : gestion des interruptions, gestion des tâches. Les fonctions de plus haut niveau, gestion de la mémoire, des fichiers se trouvent dans les couches supérieures du système.

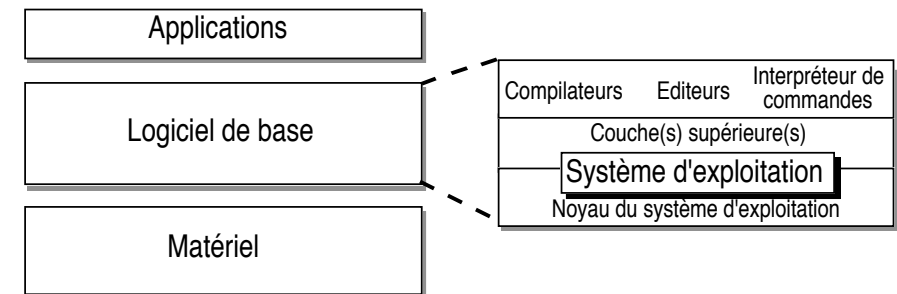
Il existe des micro-noyaux (*micro-kernels*) : on rejette à l'extérieur du noyau, et même à l'extérieur du système, tout ce qui n'est pas mécanisme de base. Par exemple, la gestion de la mémoire en mémoire virtuelle, travail particulièrement lourd, devient une option qui peut ne pas être choisie si la vitesse d'exécution est un critère primordial.

Un gestionnaire de fichiers répartis offre la possibilité à chaque utilisateur de l'un des sites d'un réseau de voir les fichiers implantés sur les autres sites. Ce gestionnaire fonctionne suivant le principe du client/serveur : un site est client de ceux qui lui offrent leurs fichiers. Ceux-ci proposent un service, ils s'appellent donc serveurs.

Introduction

Organisation d'un système

- Couches de logiciel :



Introduction

1. Rôle d'un système d'exploitation
2. Historique
3. Caractéristiques
- + Exemple : UNIX**
5. Interface utilisateur
6. Comment comparer: mesure des performances.

Introduction

Le système UNIX est organisé en deux couches : kernel (K) et user (U), à l'origine, tout ce qui a pu être retiré du noyau l'a été. Le noyau inclut la gestion des fichiers, de la mémoire et des processus (tâches). Les compilateurs, interprètes de commandes (shells), éditeurs, etc font partie de la couche U. Le noyau est presque entièrement écrit en C (1000 lignes d'assembleur pour 10000 lignes de C), ceci était complètement nouveau dans les années 70 et a permis une diffusion rapide du système.

L'une des raisons du succès du système Unix est qu'il propose de très nombreux outils de communications entre machines. Ces outils permettent l'accès aux ressources de tous les sites d'un réseau pour chaque utilisateur de l'un quelconque de ces sites.

L'envers de cette grande qualité, l'ouverture, est son faible niveau de sécurité, c'est-à-dire à la résistance aux attaques.

Les commandes de bases du système UNIX ont été présentées dans le module stations de travail.

Introduction

Exemple : Le système UNIX

- Développé par *Thompson* et *Ritchie* en 1969-70 aux laboratoires BELL (ATT).
- Dérivé de *CTSS* et de *MULTICS* (MIT).
- Principales caractéristiques :
 - deux couches : kernel et user,
 - très grande **portabilité** : noyau écrit en C à 90 %,
 - **nombreux outils de communications**,
 - toutes les ressources sont vues comme des fichiers,
 - banalisation des entrées/sorties,
 - langage et interpréteur de commande (*shell*) puissant qui permet d'enchaîner facilement des commandes (*pipe*).

Introduction

1. Rôle d'un système d'exploitation
2. Historique
3. Caractéristiques
4. Exemples : Unix et MS-DOS,
- + **Interface utilisateur**
6. Comment comparer: mesure des performances.

Introduction

Langages de programmation :

- procéduraux : C, Ada, Fortran,
- fonctionnels : Lisp,
- logiques : prolog, Smalltalk
- langages objets : Java, Eiffel, C++, Ada9X

Les langages de commandes sont interprétés :

- le shell de MS/DOS,
- les nombreux shells d'Unix

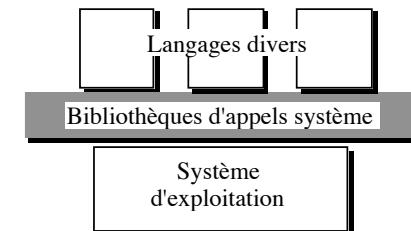
Bibliothèques, proposées par le langage (Java et Ada par exemple) ou par le UNIX pour le langage C) :

- pour gérer les entrées-sorties : C → `printf("Hello world\n")`,
Ada → `putline("Hello world");`
- pour la gestion des tâches : C → `fork();` Ada → `task`, Java → `thread`
- etc, interface graphique de Java

Introduction

Interface utilisateur : Comment décrire un travail

- Les tâches à effectuer sont décrites par un programme,
- Ce programme est écrit en utilisant soit:
 - un langage de programmation,
 - un langage de commande (`shell`),
 - des outils de plus haut niveau : générateur de programme, SQL,
- Bibliothèque des *appels système* :
 - ensemble de procédures ou de fonctions que l'on peut appeler depuis un langage de programmation pour gérer les entrées-sorties, la mémoire, les tâches, ...



Introduction

Outils de développement : voir le chapitre traitant de la chaîne de production d'une application.

Introduction

Interface utilisateur : effectuer un travail

- Outils de développement :
 - éditeurs de texte,
 - outils de maintenance (`make`), de version (`cvS`),
 - outils de mise au point (débugueurs, traces graphiques...)
- Chaîne de production d'un programme :
 - écrire le programme (fichier source),
 - compiler puis produire un fichier exécutable
 - exécuter ce dernier fichier.
- Environnement de multifenêtrage graphiques

Introduction

1. Rôle d'un système d'exploitation
 2. Historique
 3. Caractéristiques
 4. Exemples : Unix et MS-DOS,
 5. Interface utilisateur
- + **Comment comparer: mesure des performances.**

Introduction

Ce qu'il faut retenir de la mesure des performances :

- la puissance des processeur, souvent mise en avant par les vendeurs de processeurs ou d'ordinateurs, donne une **indication** sur la capacité de calcul maximale du processeur, rarement atteinte dans les applications courantes,
- le déroulement d'une application met en œuvre la mémoire, les disques, le réseau, etc.
- c'est le maillon le plus lent de la chaîne des ressources utilisées qui déterminera le niveau de performance de l'application
- les MIPS ne sont pas fondamentaux dans une application telle que Word, Netscape, mais plutôt la puissance des moyens d'entrées-sorties et de communications en général.

On peut faire la comparaison suivante : estimer une machine uniquement en fonction des seuls MIPS ou MFlops c'est évaluer un véhicule seulement en fonction de la puissance de son moteur, un tracteur de 200 CV et une voiture de sport de 200 CV seront classés dans la même catégorie. Pour faire le bon choix pour les labours, mieux vaut un critère de choix avec plus de facteurs.

Introduction

Mesure des performances

- Comment comparer les performances de machines et de systèmes différents ?
 - par des mesures statistiques : *MIPS*, *MegaFLOPS*, ...
 - par des programmes spécialisés : les *benchmarks*.

- MIPS (Million Instructions Per Second) :

$$n = \frac{1}{ci \times tc} \quad \begin{array}{l} ci : \text{nombre moyen de cycles par instruction} \\ tc : \text{temps de cycle en } \mu s \end{array}$$

n indique le nombre moyen de millions d'instructions par seconde

- MegaFLOPS (Floating Point Operations per Second) :

idem pour les instructions en virgule flottante

Mesure des performances : benchmarks

- Un *benchmark* est un programme spécialisé permettant de mettre en évidence les performances d'une application, d'un service, les caractéristiques d'un système, etc. On mesure le temps d'exécution de ce programme.
- L'avantage d'un *benchmark* est qu'il prend en compte les caractéristiques du matériel, les spécificités du système d'exploitation et les performances des compilateurs.
- On citera le Dhrystone (calcul sur des entiers) et le Whetstone (calcul sur des flottants) dont les résultats exprimés en *dhrystone/s* et *whetstone/s* qui reflètent la vitesse de traitement des instructions et l'architecture du processeur mais aussi la qualité du code généré par le compilateur