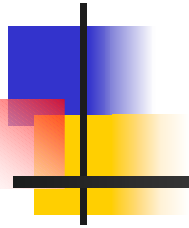


Chapitre II : Interfaces

Interpréteurs de commandes et interfaces graphiques





Interfaces

- Un SE a deux fonctions principales :
 - Gestion des ressources matérielles
 - Présentation d'une machine virtuelle à chaque utilisateur
- Nous allons détailler la seconde fonction
 - C'est une fonction des couches supérieures (proches de l'utilisateur)
 - Elle permet le dialogue entre les utilisateurs et le reste du SE : on parle d'interface utilisateur



2.1 Shells et appels systèmes

La couche **interface utilisateur** se compose de deux éléments :

- Les **shells** qui concernent tous les utilisateurs : programmeurs ou utilisateurs d'applications
- Les appels systèmes ou bibliothèques systèmes qui ne concernent que les programmeurs



2.1 Shells et appels systèmes

2.1.1 Les shells (*coquille*)

- Tous les utilisateurs emploient un shell
- C'est un langage de communication avec l'ordinateur (textuel ou graphique)
- Il masque la réalité du système à l'utilisateur
- Il représente tout ce qui est visible (ou autorisé) à l'utilisateur



2.1 Shells et appels systèmes

Exemples :

- Lorsque l'on frappe la commande `date` pour connaître la date système on utilise une fonction du shell qui sert à afficher l'heure et la date et éventuellement les modifier
- Lorsqu'on clique sur la corbeille c'est une autre fonction du shell qui est exécutée



2.1 Shells et appels systèmes

- Un shell peut prendre deux formes :
 - Interpréteur de commandes ou **CLI** (**command language interpreter**)
 - Une interface graphique ou **GUI** (**graphical user interface**)
- Dans l'exemple précédent :
 - `date` est une commande d'un shell **CLI**
 - La gestion de la corbeille fait partie de la **GUI**



2.1 Shells et appels systèmes

2.1.2 Les appels systèmes

- Les appels au système se font via une API (Application Programming Interface)
- Une API regroupe des fonctions de même nature (gestion des fichiers, primitives graphiques)
- Au niveau des fichiers on parle de bibliothèques systèmes (DLL sous WinX, lib.a ou .so sous Unix)
- Les appels au système se font depuis l'intérieur des programmes
- Les appels système de bas niveau (lecture disque, transfert réseau, etc.) sont gérés par des **interruptions logicielles**



2.1 Shells et appels systèmes

Notion d'interruption :

- Lorsqu'une machine est en marche, contrôlée par le SE, le mécanisme d'interruption permet (aux périphériques) de signaler au microprocesseur qu'il doit tenir compte d'événement extérieurs
- Lorsque le microprocesseur est occupé le mécanisme d'interruption permet de **suspendre l'exécution du programme** et de traiter un événement
- **Exemples** : une touche est enfoncée, l'imprimante n'a plus de papier, le disque a terminé sa lecture



2.1 Shells et appels systèmes

- A chaque événement correspond un certain traitement à effectuer par l'UC
- Ce travail est codé sous la forme d'une routine du SE (BIOS : Basic Input/Output System)
- Pour exécuter cette routine il faut que le processeur suspende son exécution (**rupture de séquence**) et commence à exécuter la routine
- On distingue deux types d'interruption

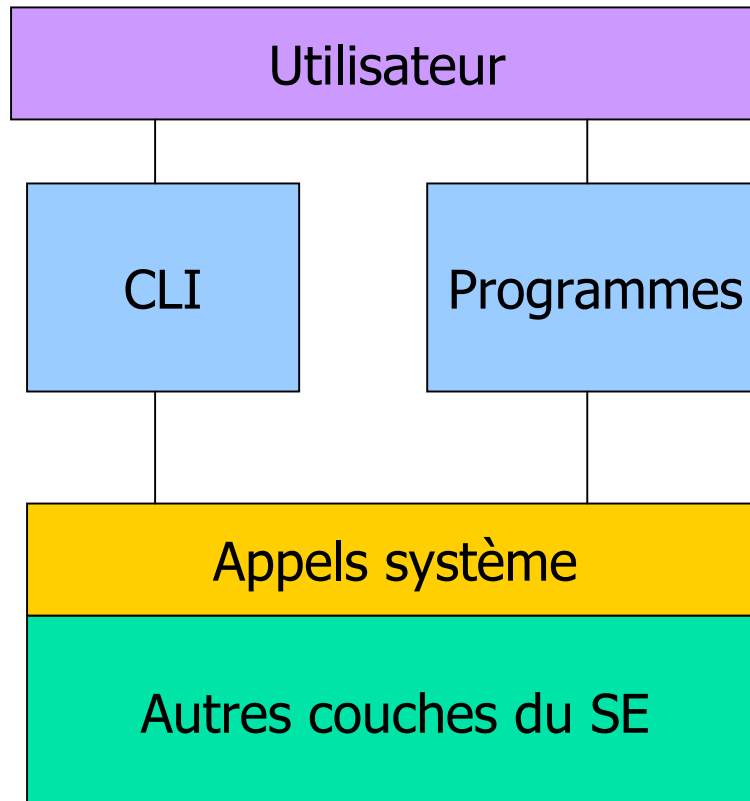


2.1 Shells et appels systèmes

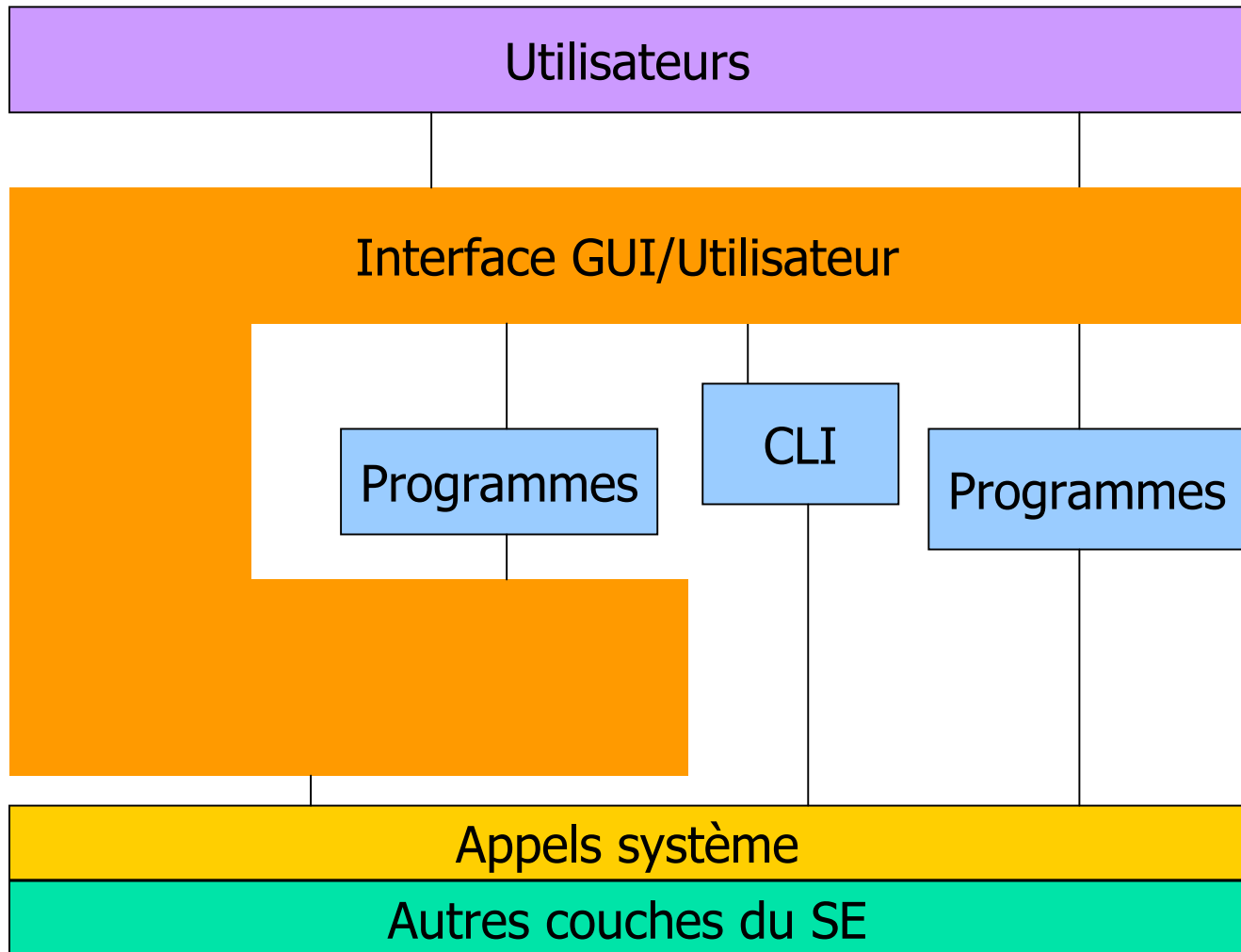
- **Interruptions logicielles :**
 - Interruptions spéciales destinées à un programme utilisateur pour faire une requête au système
- **Interruptions matérielles :**
 - Générées par les composants pour signaler un événement au processeur
- **Portée des appels au système :**
 - Tous les services ne peuvent pas être appelés par l'utilisateur :
 - **Exemple** des appels réseau ou disque

2.1 Shells et appels systèmes

2.1.3 Place du shell et des appels systèmes dans le SE



2.1 Shells et appels systèmes





2.2 CLI

- CLI = command language interpreter
- **Exemples** : MS-DOS, Shells UNIX (bash, csh, ksh etc), Console WINDOWS etc.
- La CLI ou interpréteur du langage de commande représente le système de gestion des commandes utilisateur
- **Interpréteur** :
 - agit à la manière d'un interprète humain
 - il attend une commande de l'utilisateur, l'évalue, et la traduit aussitôt en appels système (l'exécute)
 - Si plusieurs commandes se suivent, l'interpréteur les évalue séquentiellement



2.2 CLI

- **Interpréteur** opposé à **compilateur**
- Le compilateur :
 - traduit un programme en langage machine et appels systèmes
 - Mais il n'exécute pas le programme produit
 - L'exécution est différée (à la charge des utilisateur de la lancer)
 - C'est une exécution globale (plutôt que ligne à ligne)



2.2 CLI

- Le langage de commande proposé par une CLI regroupe l'ensemble des instructions que l'utilisateur peut exprimer
- Les instructions portent en général un nom qui permet de déterminer leur rôle
- **Exemples** : `date`, `help`



2.2 CLI

2.2.1 Les fonctions de la CLI

Du point de vue de l'utilisateur la CLI possède deux types de commandes :

- **Les commandes internes :**
opérations de base de la CLI (dir)
- **Les commandes externes :**
opérations annexes du SE ou programmes utilisateur (copy, format, fdisk, etc.)



2.2 CLI

2.2.2 La programmation de la CLI

- Le **langage de commande** peut s'employer comme un **langage de programmation**
- Un programme est alors une séquence de commandes placées dans un fichier textuel
- Quand l'utilisateur appelle le fichier l'interpréteur de commande les exécute successivement comme si elles avaient été tapées directement au clavier



2.2 CLI

- Sous MS/DOS, WINDOWS on parle de fichier batch
 - Ils comportent une extension (suffixe) `.BAT`
- Sous UNIX de fichier shell,
 - Ils comportent une extension généralement l'extension `.sh`
 - Ils sont rendus exécutable en changeant une de leur propriété au moyen de la commande `chmod`
 - `chmod u+x monprog.sh`



2.3 GUI

- GUI = graphical user interface
- **Exemples** : MAC OS, WINDOWS, XWINDOWS
- La GUI encapsule la couche supérieure du SE
- C'est un niveau supplémentaire qui est beaucoup plus complexe que la CLI
- **Plus simple** d'utilisation, plus ergonomique
- **Mais moins directe**, moins souple et moins efficace que la CLI pour les actions répétitives



2.3 GUI

- Une interface graphique réalise une abstraction des objets manipulés par l'OS : par exemple, les fichiers, répertoires, périphériques, programmes, commandes sont représentés sous la forme de pictogrammes appelés **icônes**
- Un système de gestion de zones permet de séparer l'écran en plusieurs parties et de distinguer les programmes actifs
- Les zones sont des **fenêtres**
- On désigne cette approche par le terme **multi-fenêtrage**



2.3 GUI

Les éléments principaux de la GUI sont :

- Un système Multi-fenêtrage
- Un système de gestion des d'événements
- Une interface d'application standardisée
- Une boîte à outils (API)
- Une langage de description et de présentation (ex : thèmes de bureau sous windows)

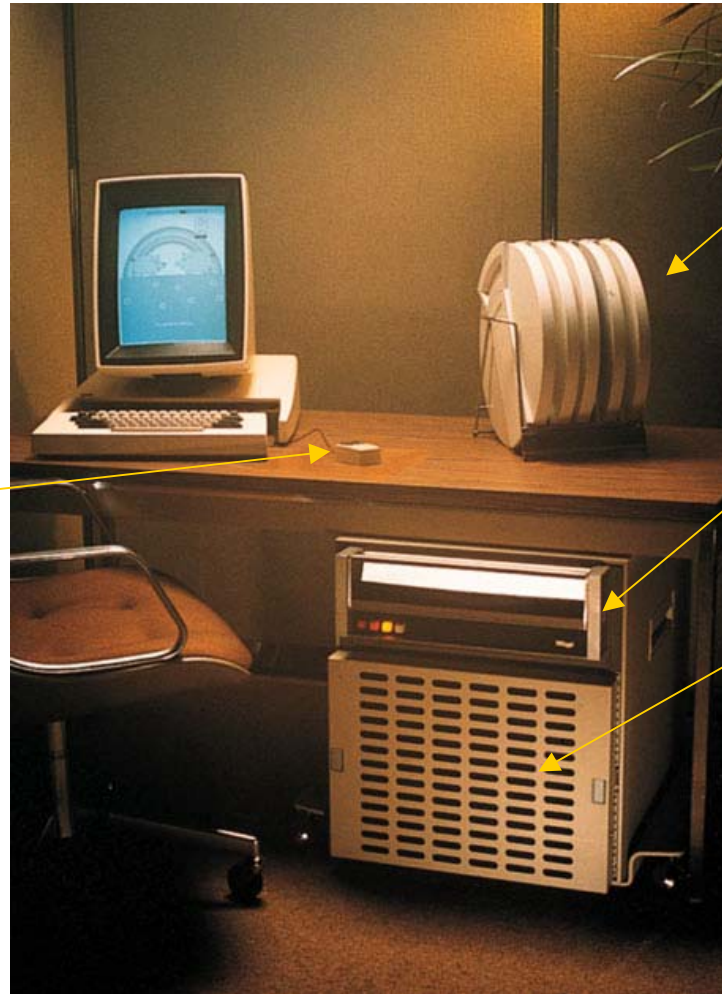


2.3 GUI - historique

- **Mars 1973** : le prototype de la station de travail **Xerox Alto** démarre pour la première fois. Son écran graphique affiche une image représentant un personnage de "1 Rue Sésame"
- La station de travail conçue au **PARC** utilise un langage orienté objet **SmallTalk**, une interface graphique, une **souris** et peut être mise en réseau au moyen d'une interface Ethernet
- **Avril 1973** : le **Xerox PARC** met au point une machine dotée de la première carte graphique couleur capable d'afficher une image de 640x486 en 256 couleurs et aussi de numériser un signal vidéo.
- Réalisation du programme **Superpaint** qui est à la fois un logiciel de dessin en couleurs et aussi le premier logiciel d'effets vidéo numériques.

2.3 GUI

Xerox Alto

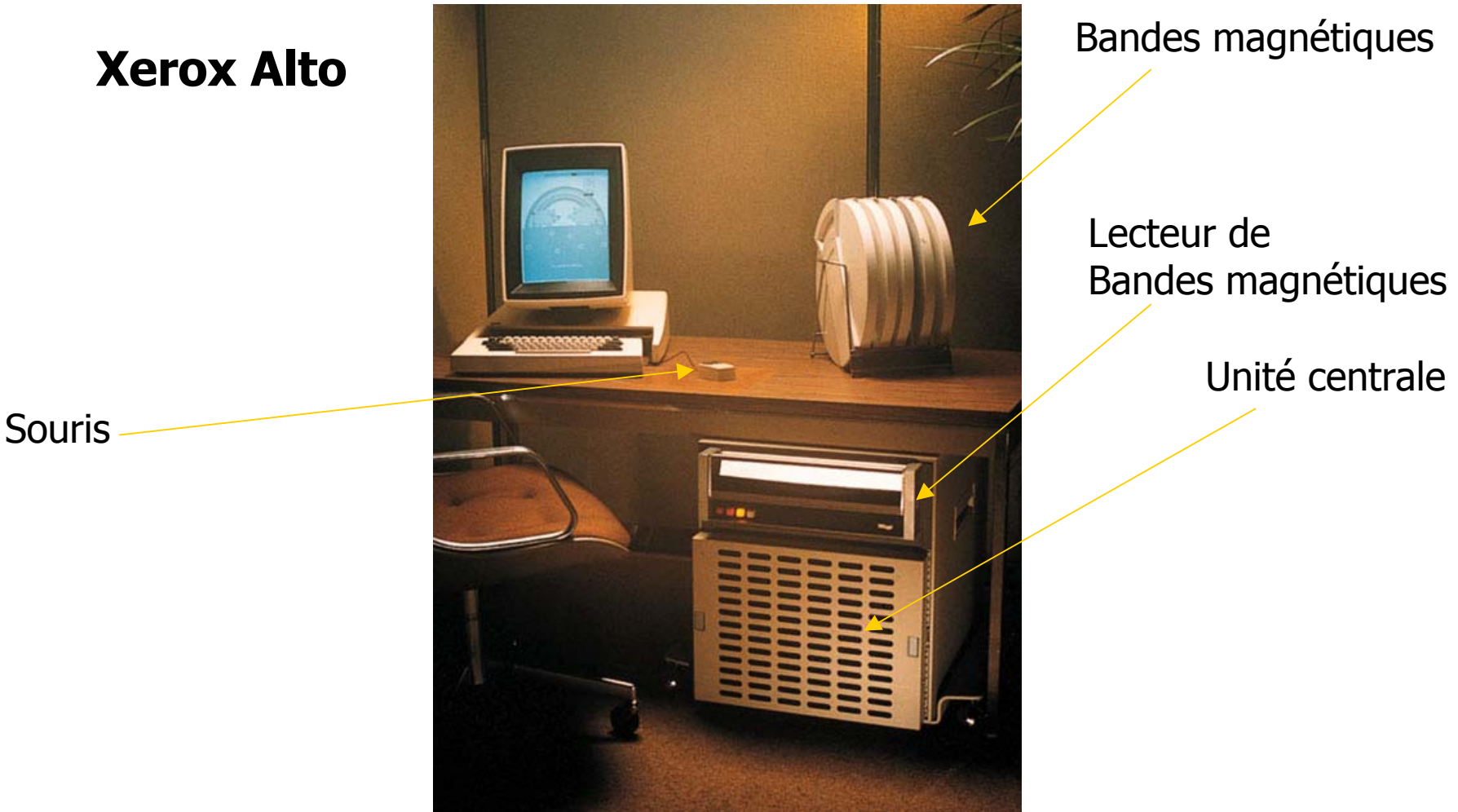


Bandes magnétiques

Lecteur de
Bandes magnétiques

Unité centrale

Souris





2.3 GUI

- **Février 1975** : Le premier traitement de texte **WYSIWYG** (What You See Is What You Get) : **Bravo** est développé au **PARC** sur **Xerox Alto**
- **Décembre 1979** : Un groupe de développeurs de chez **Apple** dont **Steve Jobs** assiste à une démonstration de l'**Alto**.
- Ils avaient été invités par **Xerox** qui, ayant investi dans la toute jeune société **Apple**, souhaitait leur montrer ce qui était en cours de développement au **PARC**.

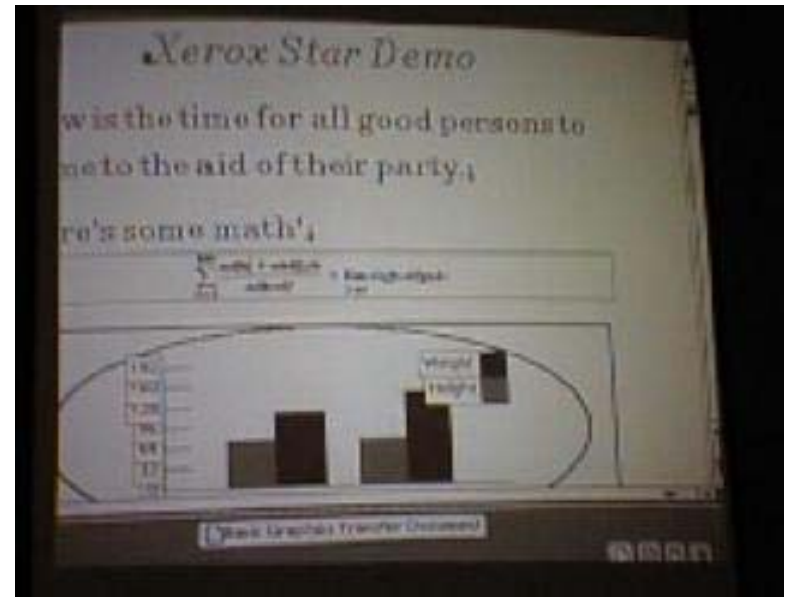


2.3 GUI

- **Avril 1981** : pour essayer de tirer parti de toutes les bonnes idées mises au point avec [l'Alto](#), **Xerox** commercialise le **Star 8010** une machine dotée
 - d'origine de 1 Mo de Ram,
 - de 8 Mo de disque dur,
 - d'une interface Ethernet,
 - d'un écran graphique,
 - d'une souris deux boutons,
 - d'une imprimante laser
 - d'une interface entièrement graphique utilisant au maximum le "Drag&Drop", le copier-coller et les menus contextuels !
- Par exemple, sauvegarder un fichier ne se faisait pas en ouvrant une boîte de dialogue "enregistrer" comme sur MacOS ou Windows mais systématiquement par Drag&Drop de la fenêtre du document vers la fenêtre contenant les icônes représentant les fichiers sur le disque. L'impression se faisait également par Drag&Drop de la fenêtre du document vers l'imprimante.

2.3 GUI

- Le **Star** était une machine en avance sur son temps
- Il a fallu plusieurs années à **Apple** et encore plus à **Microsoft** pour avoir un équivalent fonctionnel.



Ce que Steve JOBS a vu au Xerox Parc

2.3 GUI



Apple LISA (ancêtre du Mac)
1983



Macintosh 1984



2.3 GUI

- L'approche suivie par Apple fait figure de modèle
- Les principes de la GUI sont :
 - Une action donnée doit toujours avoir le même résultat
 - Un résultat donné peut avoir pour cause plusieurs actions
 - Uniformité du comportement des applications
 - Application pendant le TP n°1

2.4 Environnement multi utilisateurs

- Avant d'obtenir un shell, chaque utilisateur doit d'identifier auprès du système : processus de login
- Un utilisateur est caractérisé par :
 - Signature en entrée (authentification) composée de :
 - **Login** (nom de connexion)
 - **Password** (Mot de passe)
 - Un environnement càd lancement du shell ou de l'interface graphique et des paramètres personnels

2.4 Environnement multi utilisateurs

- Différents types d'utilisateurs
 - Super-utilisateur (root, administrateur)
 - Utilisateur simples
 - => Gestion de privilèges et des droits