

# Éléments d'histoire de l'informatique

Sacha Krakowiak

Université Grenoble Alpes & Aconit

## 8. Histoire des systèmes d'exploitation

[CC-BY-NC-SA 3.0 FR](#)

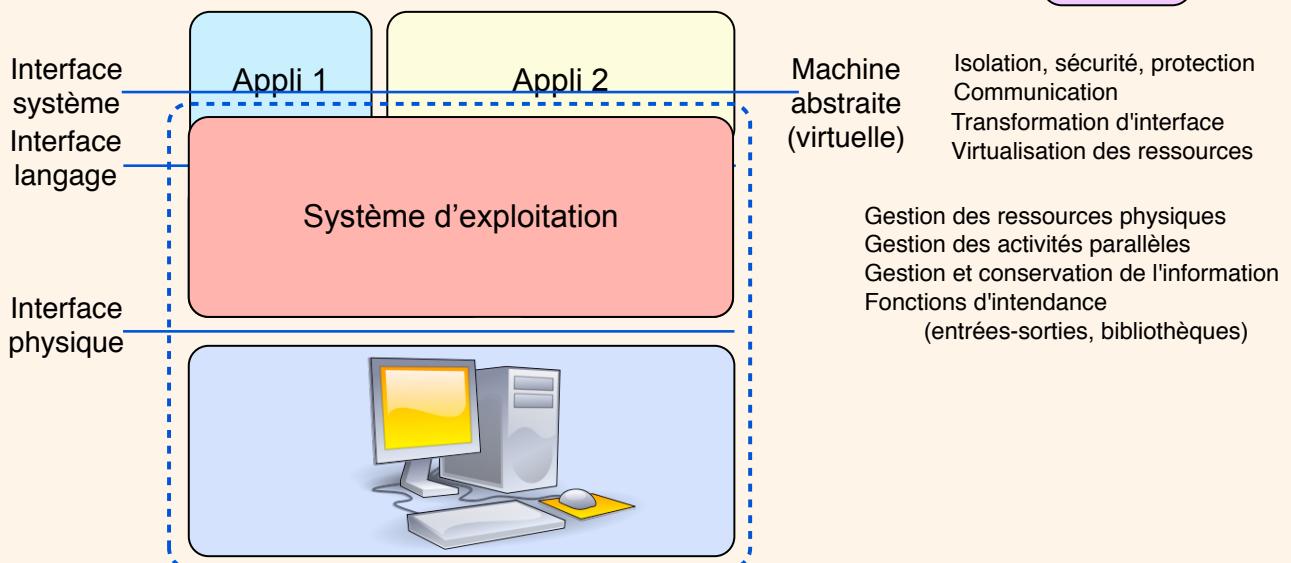
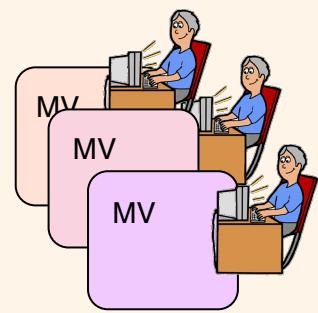
Qu'est-ce qu'un système d'exploitation ?

*“An operating system is a collection of things that don’t fit into a language. There shouldn’t be one.”*

*« Un système d’exploitation, c’est une collection de choses qui ne tiennent pas dans un langage. Ça ne devrait pas exister. »*

Dan Ingalls, “Design Principles Behind Smalltalk”, *Byte Magazine*, August 1981.

# Qu'est-ce qu'un système d'exploitation ?



## Qualités d'un système d'exploitation

### ❖ Discréction

assurer ses fonctions, et ...se faire oublier

### ❖ Économie

bon usage des ressources

performances

    temps de réponse

    débit des travaux

    surcoût minimal

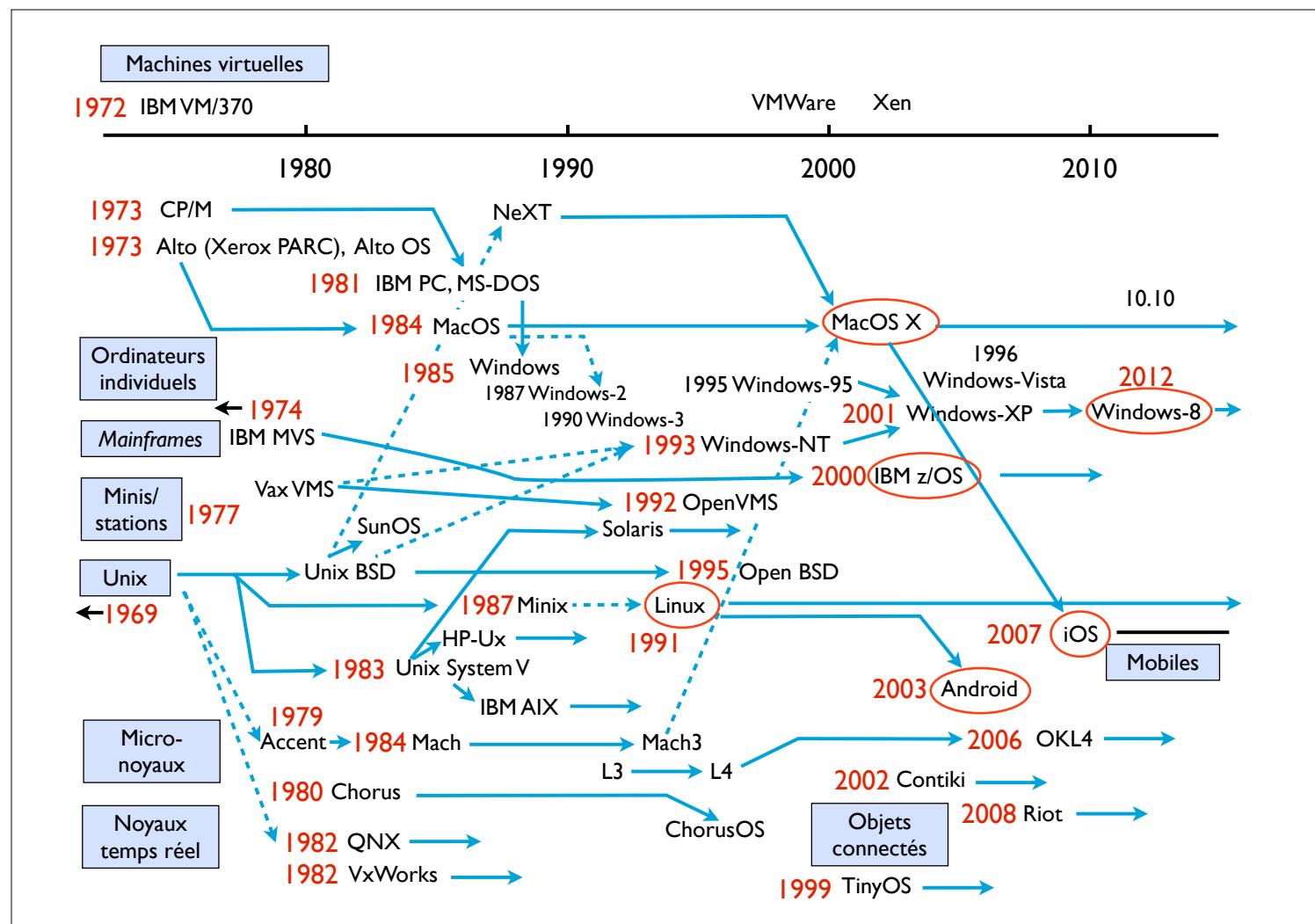
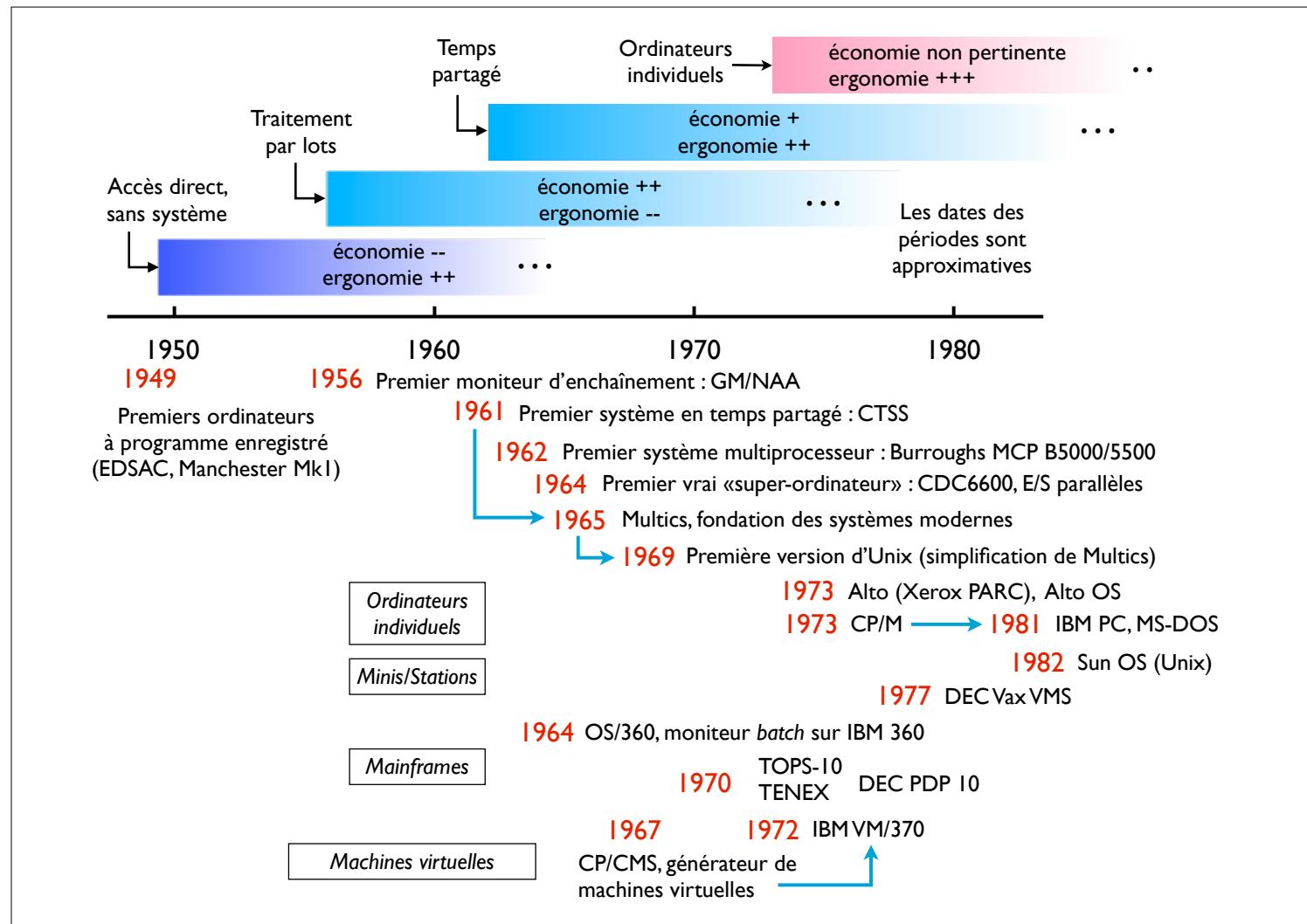
    équité

### ❖ Ergonomie

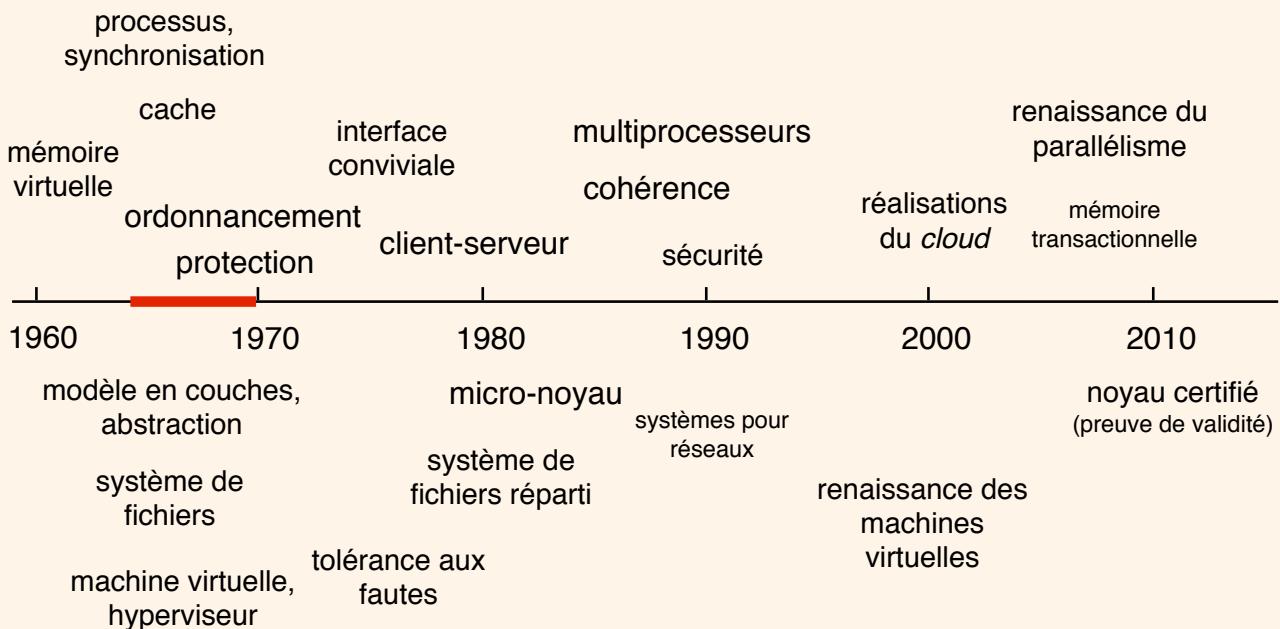
facilité d'usage

qualité de l'interface

prévisibilité



# Concepts et techniques



## La préhistoire (1950-56)

### ❖ Pas de système d'exploitation !

applications : recherche, calcul scientifique, puis gestion

### ❖ Fonctionnement sur réservation

le programmeur a le contrôle complet de la machine

programmation en binaire ou assembleur

mise au point interactive (clés)

quelques outils simples

assembleur et chargeur rudimentaires

bibliothèque de sous-programmes

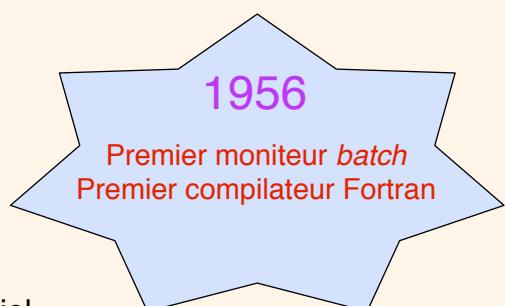
outils de mise au point

(arrêt sur adresse, affichage)

### ❖ Bon confort, mais...

peu économique, vu le coût élevé du matériel

faible productivité, vu la pauvreté des outils



# Systèmes de traitement par lots

## ❖ Motivation

rentabiliser l'investissement en matériel  
maximiser le taux d'utilisation des ressources

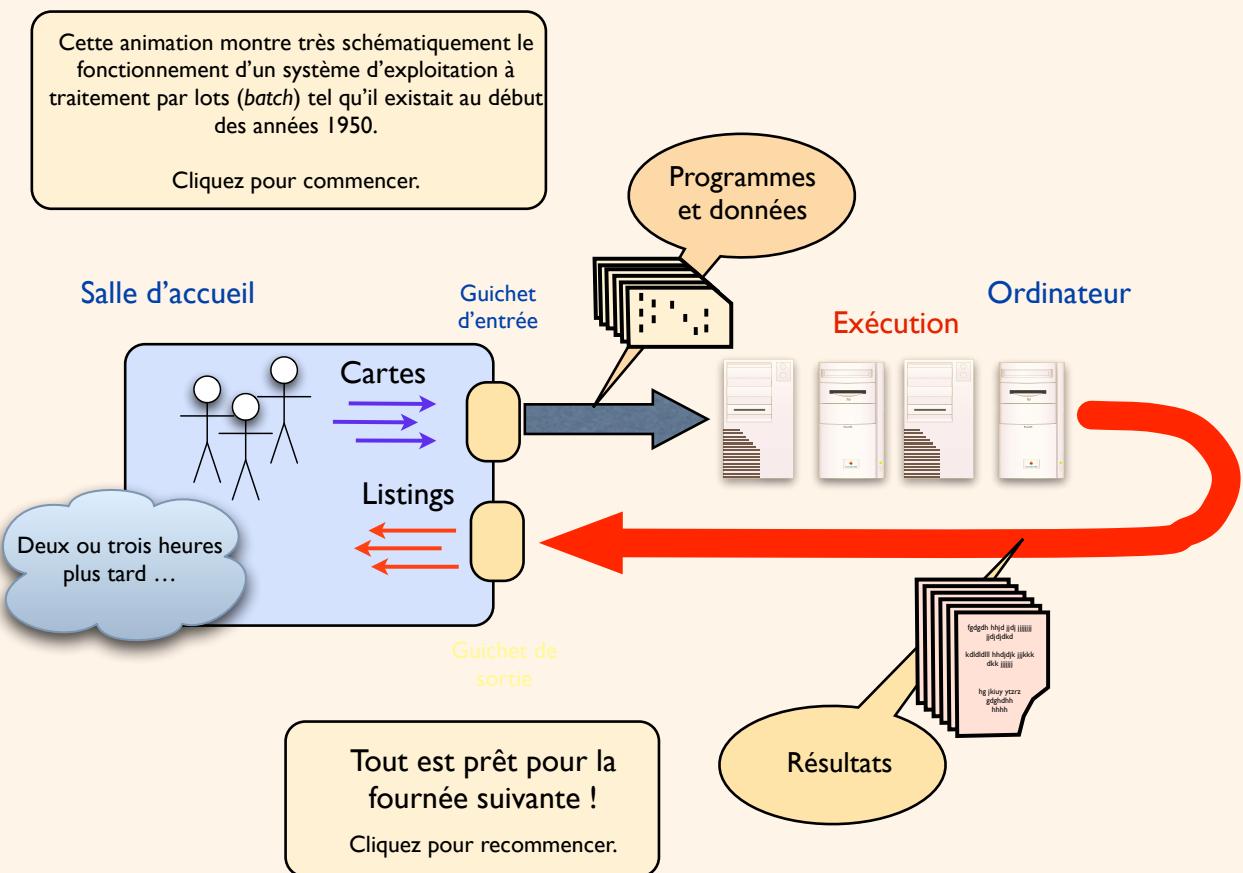
## ❖ Premier système (1956)

développé par des utilisateurs (General Motors, North American Aviation)  
sur IBM 704  
IBM suivra avec IBSYS

## ❖ Principe

regrouper les travaux par «fournées» (*batch*)  
enchaînement automatique des travaux  
entrées sur cartes, sortie sur imprimante  
plus tard : bandes magnétiques

## Moniteur de traitement par lots (*batch*)



# Un nouveau modèle d'exploitation

## ❖ L'utilisateur éloigné de la machine...

perte de l'interactivité  
longs délais d'attente  
faible tolérance aux erreurs



© FEB - Jean Bellec

## ❖ De nouveaux métiers

atelier de préparation des travaux  
suite des ateliers de mécanographie  
«perfo - vérif»  
gestion et surveillance de l'exécution  
opérateur - pupitreur



Image courtesy Computer History Museum

# Avancées architecturales

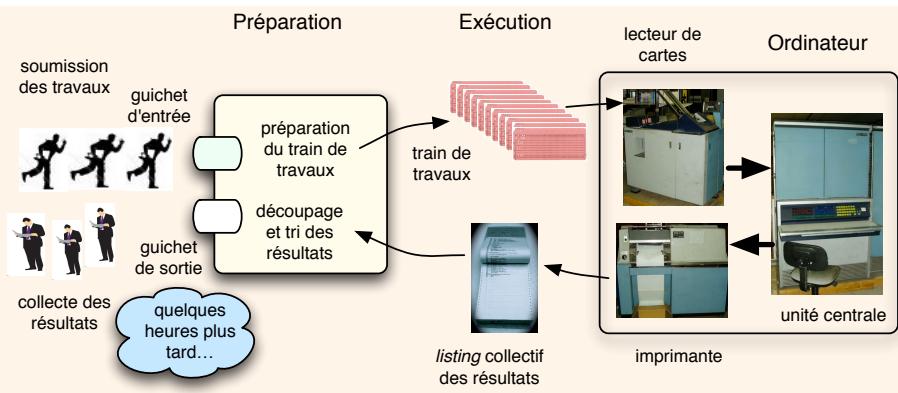
Le système d'exploitation impose des changements dans l'architecture des machines

## ❖ L'horloge programmable

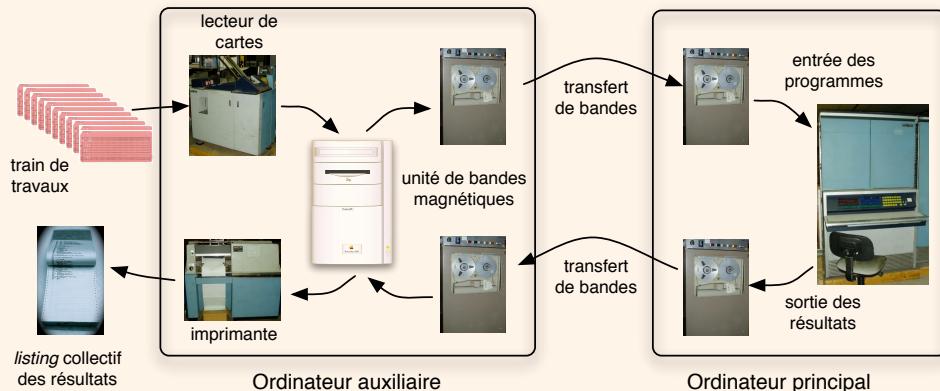
problème : boucle infinie dans un programme d'utilisateur  
 bloque tout le train de travaux  
remède : l'horloge programmable  
 chaque travail indique une durée maximale d'exécution  
 il est interrompu si cette limite est atteinte

## ❖ La protection de mémoire

problème : un programme écrit «n'importe où» dans la mémoire  
 risque d'écrire dans le programme ou les données du système  
remède  
 placer le système dans une zone de mémoire protégée



## La course à l'efficacité (1) : ordinateurs couplés



L'ordinateur principal et l'ordinateur auxiliaire fonctionnent en parallèle et peuvent être en des lieux différents.

## La course à l'efficacité (2) : La multiprogrammation

### • Objectif

réduire le temps de commutation entre travaux

### • Principe

plusieurs travaux coexistent en mémoire  
chargement des travaux en parallèle avec l'exécution

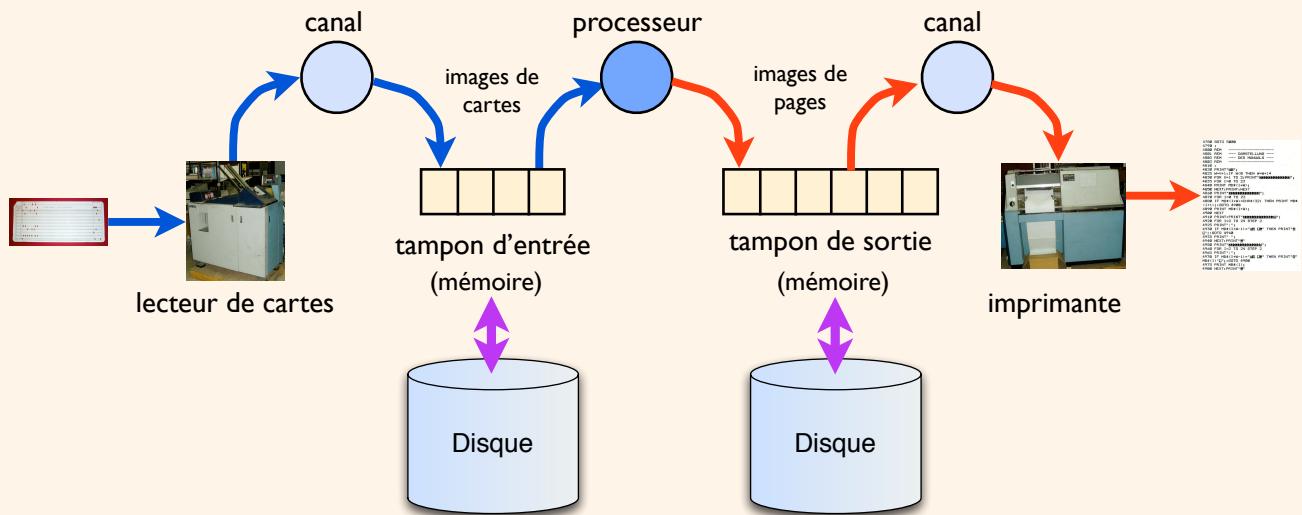
### • Influence sur l'architecture

mémoire secondaire rapide (disques)  
canaux d'entrée-sortie (E/S simultanées)  
registres de base

## Entrées-sorties simultanées

### ❖ Le *Spool* (*Simultaneous Peripherals Operation On Line*)

recouvrement entre exécution de programme et entrées-sorties  
exploite les canaux d'entrée-sortie et les disques



## Trois systèmes du début des années 60

### ❖ Deux systèmes novateurs (1961-62)

Burroughs MCP (*Master Control Program*) pour B5000/5500

premier système pour multiprocesseur

programmé en langage de haut niveau (sous-ensemble d'Algol 60)

organisation originale de la mémoire

(segments gérés par logiciel, exécution sur une pile)

le système d'exploitation est lui-même segmenté

Atlas (Univ. Manchester - Ferranti)

mémoire «à un niveau» : invention de la mémoire virtuelle paginée

première mise en œuvre du *spool*

invention des appels au superviseur

### ❖ Un système à large diffusion : OS/360 (1964)

Système d'exploitation unique pour la gamme IBM/360

# Les débuts du temps partagé

## ❖ Motivations

retrouver l'interaction directe, perdue avec le traitement par lots...  
... en maintenant une bonne exploitation des ressources

## ❖ Une vision prophétique

*"... computing may someday be organized as a public utility just as the telephone system is a public utility"*

John McCarthy (1961)

## ❖ Principe de base

Exploiter la différence d'échelle de temps entre l'homme (seconde) et le processeur (microseconde)

Pendant le temps de réflexion d'un usager, on peut servir beaucoup d'autres usagers

## ❖ Un prototype de recherche : CTSS (MIT), 1961

## CTSS (*Compatible Time-Sharing System*)

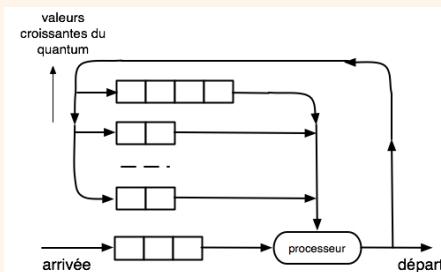
### ❖ Un système novateur

premier système en temps partagé  
(compatible avec *batch*)

ordonnancement multi-niveaux

communication entre utilisateurs

langage de commande



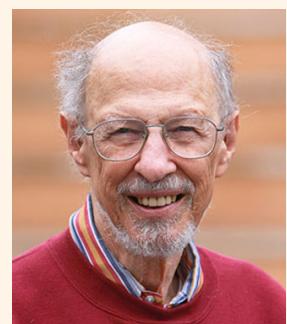
### ❖ La réalisation

IBM 7094, modifié à la demande

2 bancs de mémoire de 32K mots

protection de mémoire, système de déroutements  
(«trappes»)

version 0 : 1961 (4 terminaux), production : 1964-74



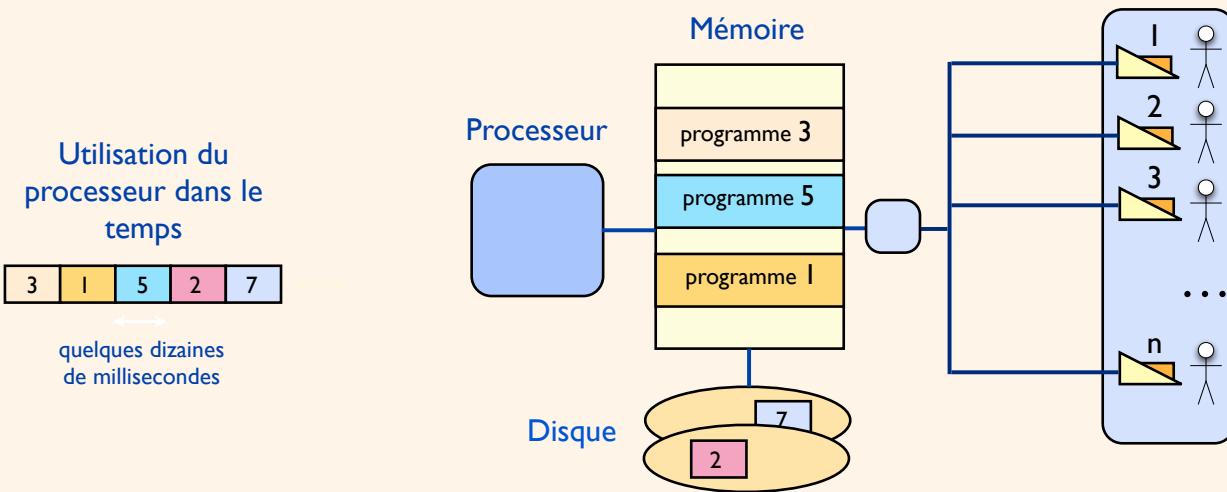
Fernando Corbató  
Jason Dorfman, MIT CSAIL

### ❖ Les retombées

démonstration de la viabilité du temps partagé

influence directe : Multics (MIT), CP/CMS (IBM)

## Principe du temps partagé



Grâce au partage du processeur :

Chaque utilisateur a l'impression qu'il dispose seul de l'ordinateur

Grâce à l'utilisation du disque :

Chaque utilisateur a l'impression qu'il dispose d'une mémoire pratiquement illimitée  
(mémoire virtuelle)

## Les débuts d'une approche scientifique des systèmes d'exploitation

### ❖ Une période charnière : 1965-70

1967 : *First ACM Symposium on Operating Systems Principles*

### ❖ Un concept clé : la virtualisation

processeur	processus	
mémoire	mémoire virtuelle	segment
disque	fichier	
entrée-sortie	flot	
écran	fenêtre	
machine	machine virtuelle	
...	...	

### ❖ Des expériences fondatrices

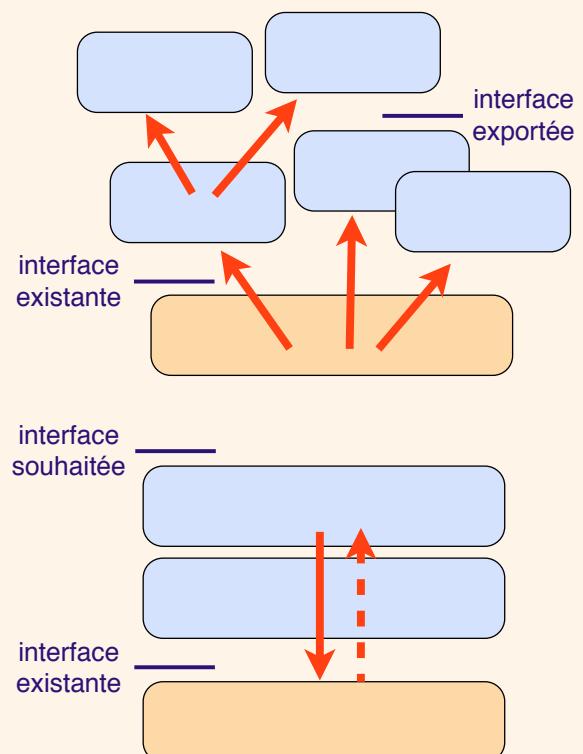
THE, Multics, CP67

# Les deux faces de la virtualisation

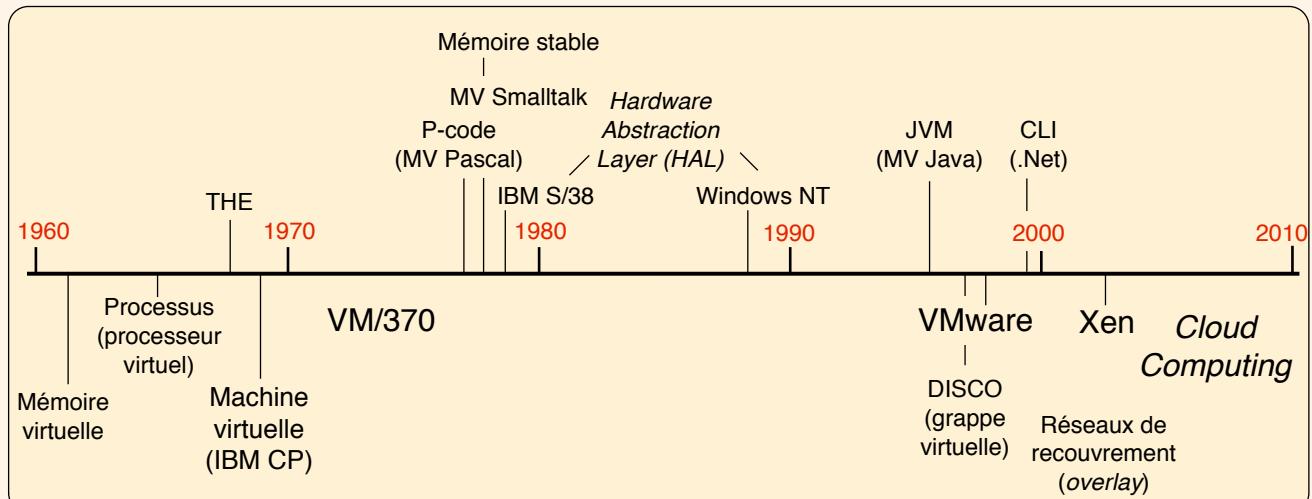
## ❖ Ascendante (abstraction)

- Un outil de partage de ressources
- Création de ressources «hautes»
- Multiplexage de ressources «basses»

Interface visible, réalisation cachée  
Transformation (ou non) d'interfaces  
Préservation d'invariants



## Brève histoire de la virtualisation



# La virtualisation

## clé des systèmes d'exploitation

### ❖ Virtualisation du processeur

La notion de processus (Edsger Dijkstra, 1965)

Les bases de la synchronisation

des sémaphores aux moniteurs

maîtriser les interruptions

### ❖ Virtualisation de la mémoire

La première mémoire virtuelle paginée (Atlas, Tom Kilburn, 1962)

La segmentation

Bob Barton, Burroughs B5000, 1961

Jack Dennis, Multics, 1965

Maîtriser l'écroulement (1968)

localité et *working set*

régulation

### ❖ Virtualisation des entrées-sorties

Flots d'entrée-sortie et périphériques virtuels

## Bilan des débuts de la programmation concurrente 1965-75

### ❖ Avancées ...

La notion de processus

Le problème de l'exclusion mutuelle et ses solutions

Compréhension et solution de l'interblocage

Des outils de synchronisation : sémaphores, messages, moniteurs

### ❖ Limites ...

Preuve de programmes parallèles

première réponse en 1976 (Owicki-Gries)

Intégration dans les langages de programmation

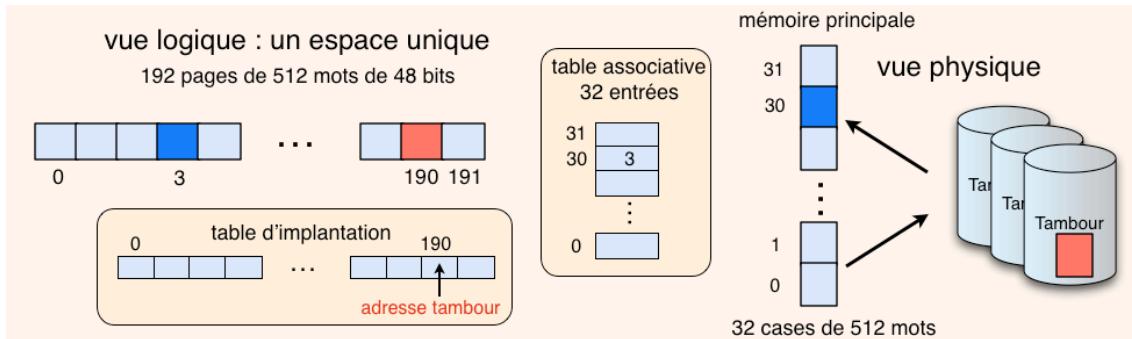
premières réponses à partir de 1975 (Concurrent Pascal, Mesa, ...)

Prise en compte des défaillances

transactions, à partir de 1976

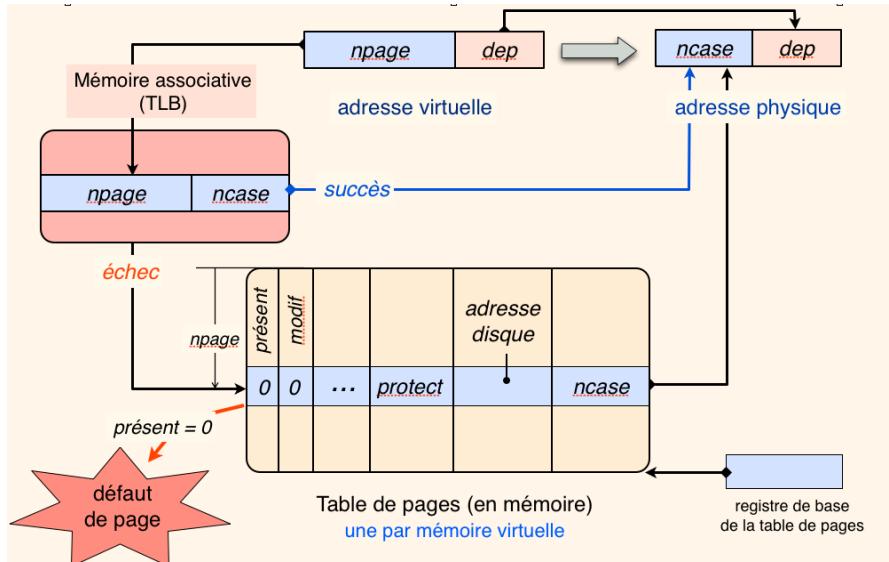
Prise en compte de la répartition

horloges logiques, 1978



## Réalisation de la mémoire virtuelle

Actuel  
(très simplifié)



## Machines virtuelles

### L'hyperviseur, un super-OS

Présente une interface uniforme aux machines virtuelles (MVs)

Gère (et protège) les ressources physiques

Encapsule l'état interne des MVs

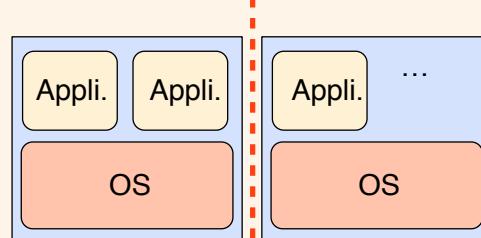
C'est un composant critique

Isolation

Défaillances

Sécurité

Interface machine physique



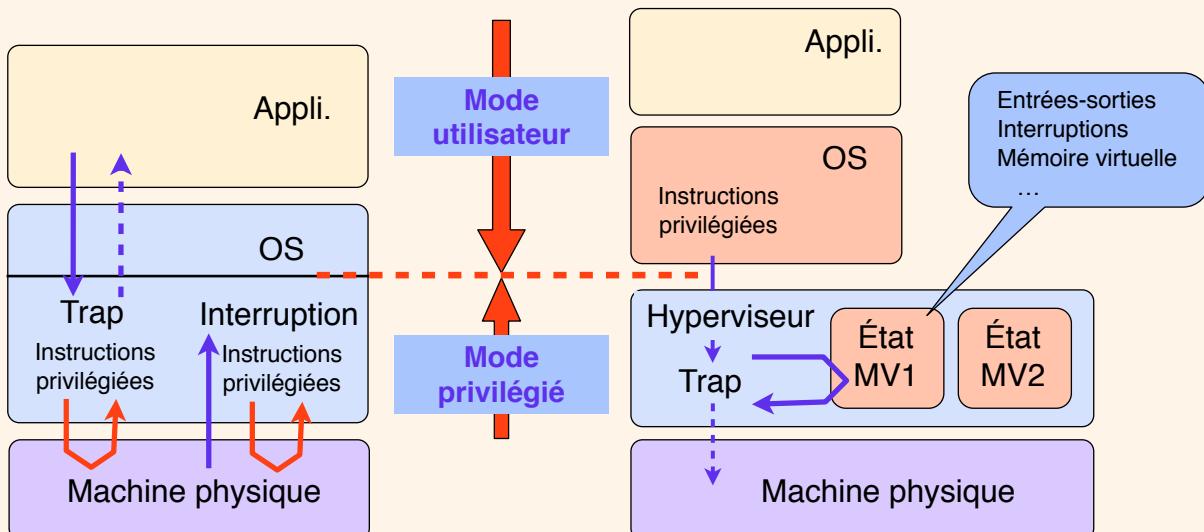
### Première réalisation

CP67 (IBM 360/67)

Utilisation : CP/CMS

Base pour VM/370

# Machines virtuelles : comment ça marche



## ❖ Un problème ...

- Sur les machines actuelles (IA-32, etc.), l'effet de certaines instructions **dépend du mode courant** (privillégié ou utilisateur)  
De telles instructions ne sont pas virtualisables

# Machines virtuelles

## ❖ Comment contourner le problème des instructions multi-modes ?

La paravirtualisation : changer l'interface de l'hyperviseur

Remplacer les instructions non virtualisables

L'interface n'est plus celle de la machine physique

Les OS doivent donc être modifiés !

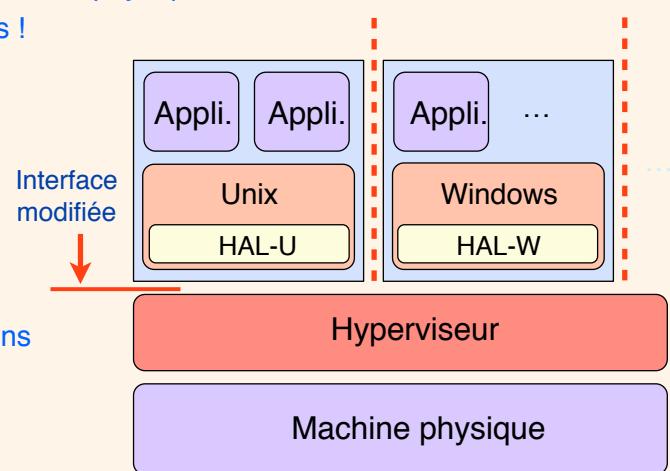
En pratique, on ne modifie que  
la HAL (couche d'abstraction)

La traduction dynamique  
de code binaire :

Remplacer à la volée les instructions  
non virtualisables

Dans l'avenir :

Les nouveaux processeurs sont conçus pour la virtualisation



## L'époque moderne (après l'avènement de l'approche scientifique)

### Multics, à la charnière de deux époques

#### ❖ Un projet ambitieux...

Suite du succès de CTSS  
Objectif : un service universel de calcul  
Consortium MIT, General Electric, Bell Labs

#### ❖ 1965-70 : le début de l'ère moderne des systèmes

Les principes sont établis (processus, mémoire virtuelle, protection, gestion de l'information)

Premier *Symposium on Operating Systems Principles* en 1967

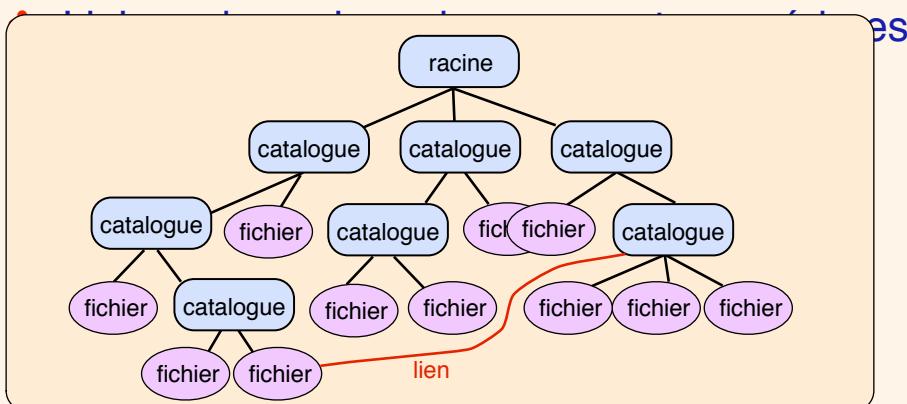
Multics intègre ces principes et joue un rôle de pionnier

#### ❖ ... mais une histoire mouvementée

Projet lourd et complexe, performances décevantes au début  
Dissolution du consortium en 1969-70  
Percée commerciale limitée et très tardive

## Multics, système fondateur

- ❖ Matériel sur mesure (GE-645, modification du GE-635 ~ IBM 7094)
  - ❖ Mémoire virtuelle segmentée (support matériel)
    - pas de distinction entre segment et fichier
    - gestion des segments par pagination à la demande
  - ❖ Modèle hiérarchique de gestion des fichiers
    - universellement adopté, avec variations



CC-BY-NC-SA 3.0 FR - S. Krakowiak, 2016/17

Histoire de l'informatique

8 -

# Vie de Multics

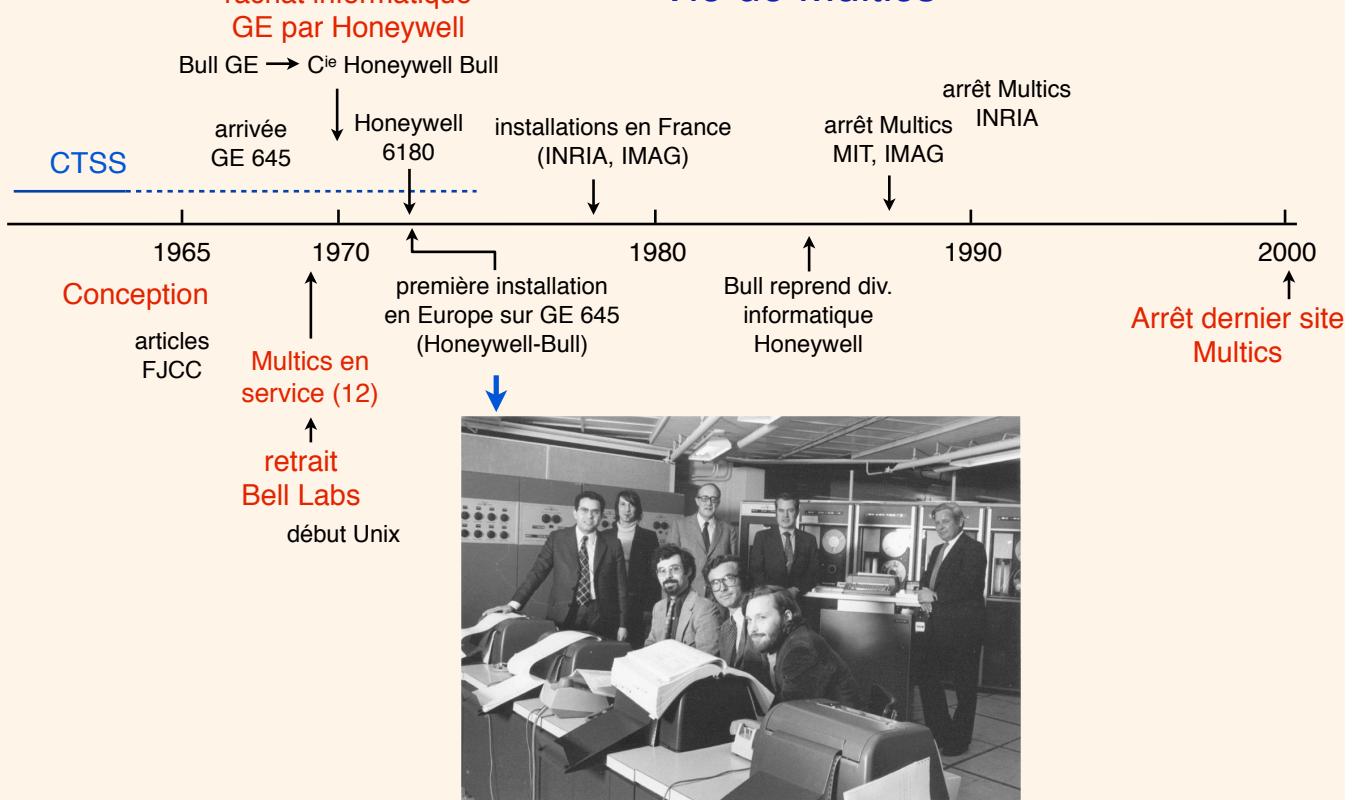


photo du GE-645 des Études Bull Paris 1972 conservée par Marc Loux  
Au premier rang Dave Vinograd, Allen Berglund (†), Rick Gumpertz  
Au second rang l'équipe : Françoise : Albert Harari, Marc Loux, François Du Jeu  
Ainsi que John Zona (Field Engineer), John Ammons (CPU Designe)

©FFEB

---

CC-BY-NC-SA 3.0 FB - S. Krakowiak 2016/17

Histoire de l'informatique

---

8 - 32

# De Multics à Unix

1969

Les Bell Labs quittent Multics

Ken Thompson et Dennis Ritchie lancent le projet Unics «en perruque» sur PDP-7

1970

Les Bell Labs commencent à s'intéresser au projet

Unics devient Unix, multi-utilisateurs porté sur PDP 11/20

1972

Ritchie réécrit Unix dans le langage C Unix devient portable Première installation commerciale

1974

Publication d'un article dans les CACM  
Début de la communauté Unix

1975-76

*"Lions' Commentary"* V6  
Expansion de la communauté  
Première licence

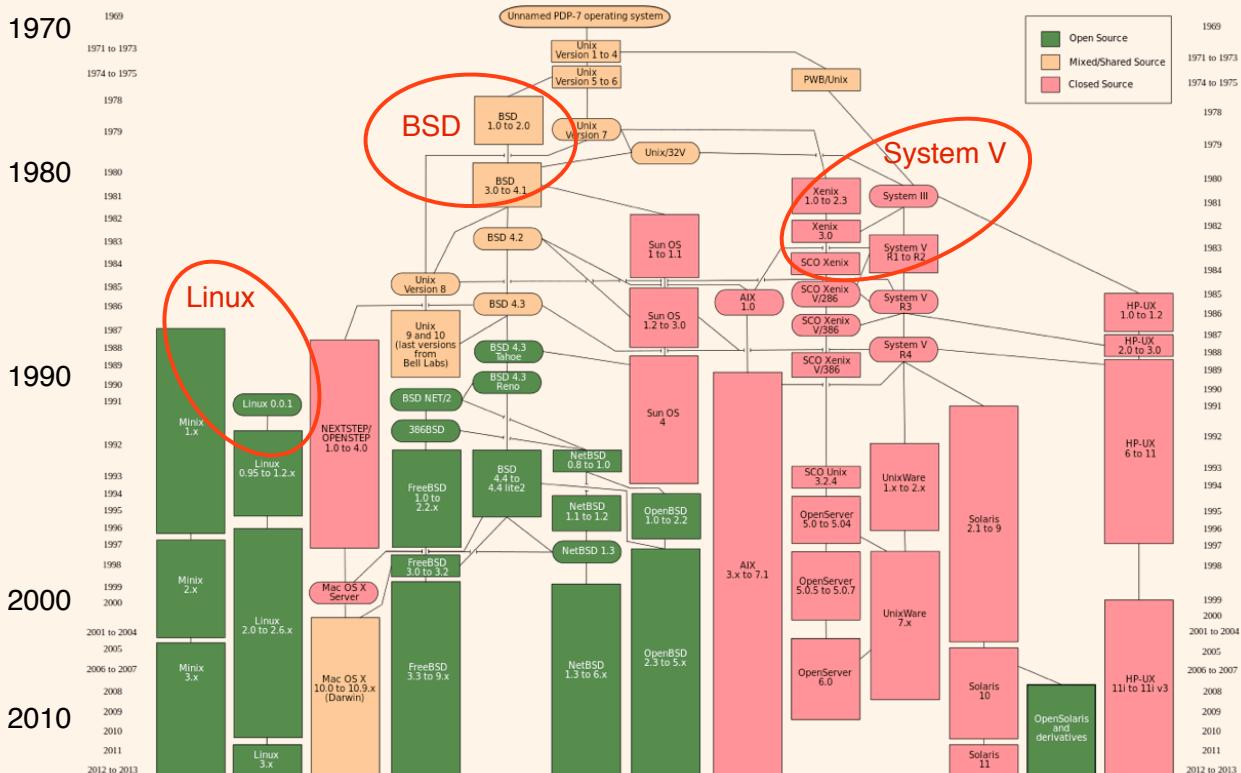
1979-80

Premières versions industrielles

1988

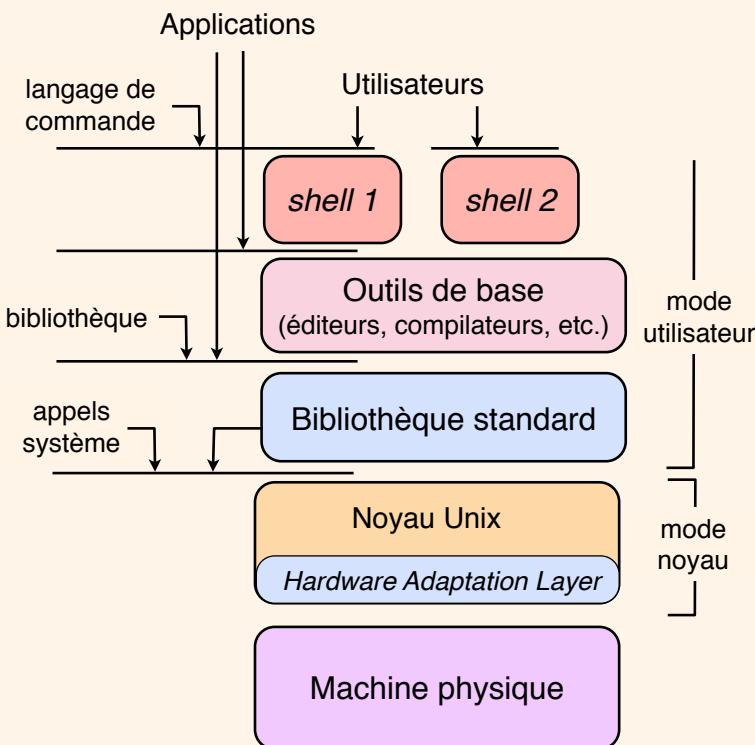
Première version de Posix

## La famille Unix



source : Wikipedia, [Levez Unix History Diagram](#)

# Unix : architecture d'ensemble



Dennis Ritchie, Ken Thompson  
Source : Wikipedia, author : Peter Hamer

## Unix : principes de base

- ❖ **Entité de base : le processus**
  - mémoire virtuelle linéaire + flots d'exécution (*threads*)
  - segments «mappés» dans la mémoire virtuelle
  - mémoire virtuelle partagée
- ❖ **Fichiers**
  - système hiérarchique à la Multics + protection simple
  - descripteur (*inode*) + tables d'implantation multiniveaux
  - outil universel de gestion de l'information (inclut entrées-sorties)
- ❖ **Allocation des ressources**
  - processeur : tourniquet multiniveaux
  - mémoire paginée à la demande avec limiteur de charge
- ❖ **Shell**
  - composition d'outils élémentaires (*pipe*, flots, redirection)

## Protection

### ❖ Un premier modèle, la matrice des droits (vers 1970)

agents \ objets	fich1	fich2	périph.	D1	D2	D3
D1	<lire, écrire, exec. >	<lire, écrire>	<allouer, réquisit. >	<>	<appel>	<changer droits>
D2	<lire, exec. >	<lire, exec. >	<demander, libérer. >	<appel>	<>	<appel>
D3	<exec. >	< nil>	<>	< nil>	<appel>	<>

### ❖ Deux lectures

par colonnes : listes d'accès

par lignes : listes de droits et capacités

## Les systèmes à capacités

### ❖ Une idée intéressante, mais qui n'a pas pris...

Une extension du principe de l'adressage segmenté

Permet d'assurer la «méfiance mutuelle», contrairement aux anneaux

Une capacité (*capability*) = une adresse + une clé de protection

Une capacité fournit un accès protégé à un «objet»

Les capacités doivent être protégées contre la copie et la contrefaçon

Des réalisations logicielles (CAL-TSS, Hydra) puis matérielles (machines à capacités), années 1970-75

Plessey 250, IBM 38, CAP, Intel iAPX 432

### ❖ Exemple : Plessey 250

Une machine pour des applications de télécommunications

Les capacités sont dans des segments spéciaux (C-listes) protégés

De même, les registres de capacités sont distincts des registres ordinaires

Notion de domaine : ensemble d'objets, chacun muni de droits d'accès

Un domaine est désigné par une liste de capacités (C-liste)

Un processus passe d'un domaine à l'autre par appel de l'un de ses points d'accès

# Micro-noyaux

## ❖ Motivations

Éliminer les inconvénients des systèmes monolithiques  
Modularité, construction de systèmes «à la carte»

## ❖ Principes

Le micro-noyau assure 3 fonctions

- gestion de mémoire à bas niveau
- gestion des activités (*threads*)
- communication par messages

Un système d'exploitation est construit à partir de ces fonctions

## ❖ Problèmes

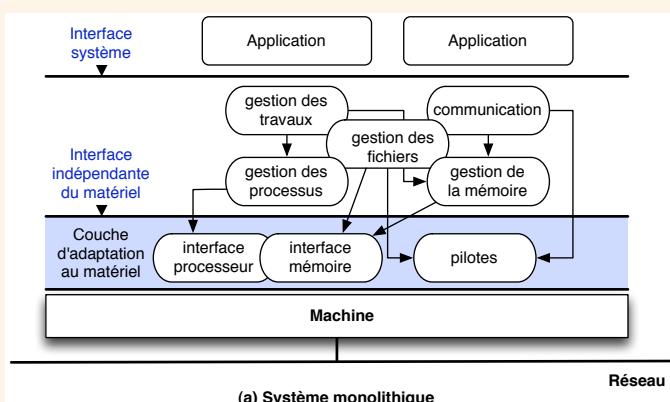
- Manque de souplesse
- «Abstractions» imposées
- Performances

## ❖ Réalisations

Mach ; Chorus ; L3, L4

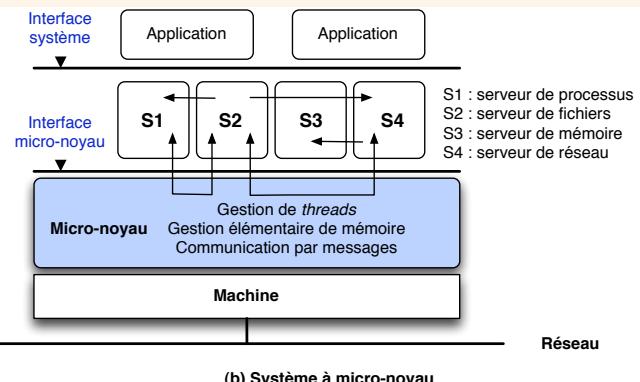
Domaine actif dans les années 1980 à 1995  
En retrait depuis, mais renouveau récent (systèmes embarqués)

## Système sur micro-noyau

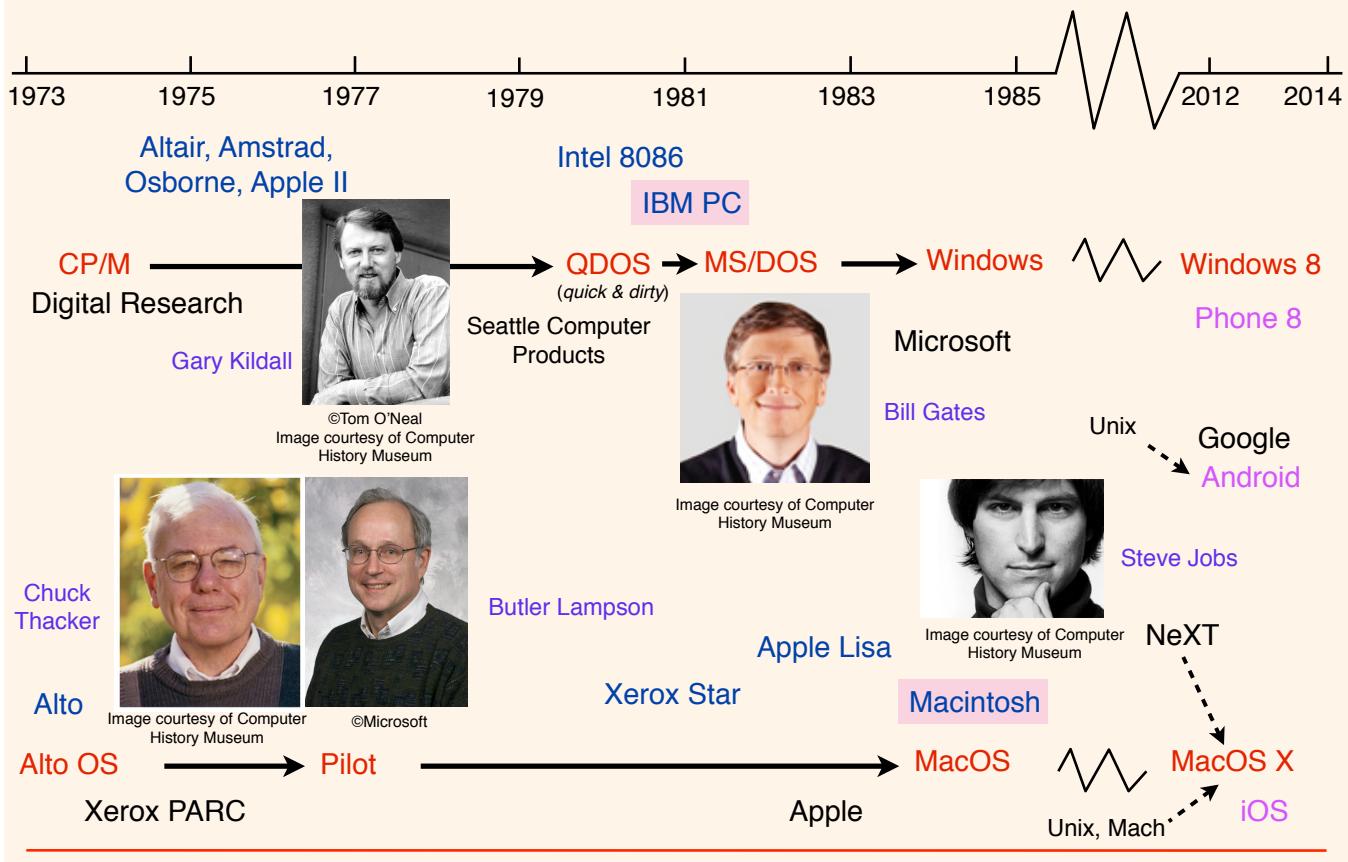


### Exemples

Windows NT  
Mac OS X (Darwin sur Mach 3)



## Les débuts des systèmes d'exploitation pour ordinateurs individuels



## Systèmes pour objets connectés

- ❖ **Domaine d'application**  
micro-contrôleurs, réseaux de capteurs
- ❖ **Contraintes**  
faible consommation d'énergie  
capacité de mémoire limitée
- ❖ **Exigences**  
fiabilité, adaptabilité, qualité de service
- ❖ **Exemple : TinyOS (U. Berkeley, Intel, Crossbow Technology)**  
système modulaire, état encapsulé  
piloté par les événements (origine externe ou interne)  
file d'ordonnancement  
tâche de fond



# Virtualisation dans les systèmes embarqués

## ❖ Contraintes spécifiques

- Complexité croissante des applications
- Maîtrise des performances
- Réduction de la taille de la base informatique de confiance

## ❖ Un renouveau pour les micro-noyaux

- Le micro-noyau comme hyperviseur à bas niveau
- Des systèmes d'exploitation «sur mesure» pour une application
  - permet de réaliser des «appareils virtuels» (appareil + son OS spécifique)
- Les pilotes peuvent sortir de la base de confiance
- Les composants critiques peuvent être isolés dans des MV

G. Heiser, "The Role of Virtualization in Embedded Systems", *Proc. First Workshop on Isolation and Integration in Embedded Systems (IIIES'08)*, pp 11-16, April 2008

## Un micro-noyau vérifié

### ❖ Une version du micro-noyau L4

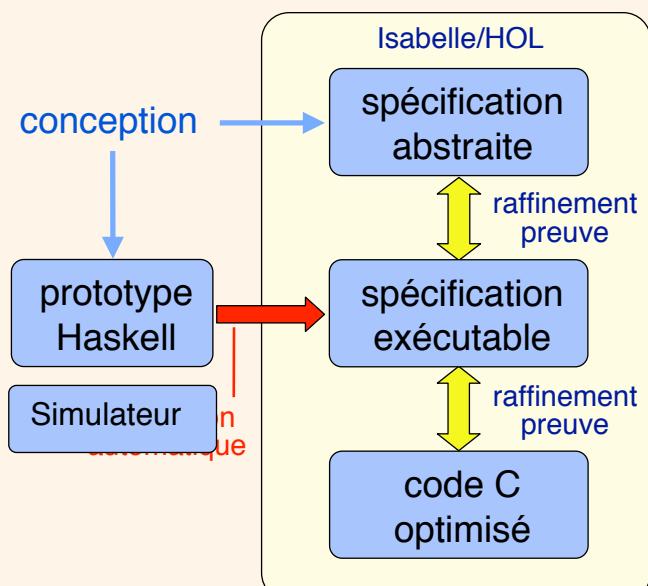
Machine virtuelle donnant une image simplifiée du matériel

### ❖ Difficultés

Asynchronisme  
Gestion de la mémoire (pointeurs)  
Accès direct aux fonctions du matériel (MMU, ...)

### ❖ seL4, base d'un “micraviseur”

OKL4 (Open Kernel Labs)  
Base installée : un milliard ...



Gerwin Klein et al., "seL4: Formal Verification of an Operating-System Kernel", *Communications of the ACM*, vol. 53, no 6, pp. 107-115, June 2010

## Les défis des systèmes d'exploitation

- ❖ Sécurité
- ❖ Certification
- ❖ Systèmes embarqués et mobiles
- ❖ Parallélisme (multi-cœurs)
- ❖ Gestion de grands volumes de données
- ❖ Virtualisation à grande échelle
- ❖ Autonomie et adaptation