Histoire des systèmes d'exploitation

Sacha Krakowiak

Université de Grenoble & Aconit

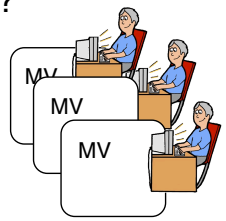
Qu'est-ce qu'un système d'exploitation ?

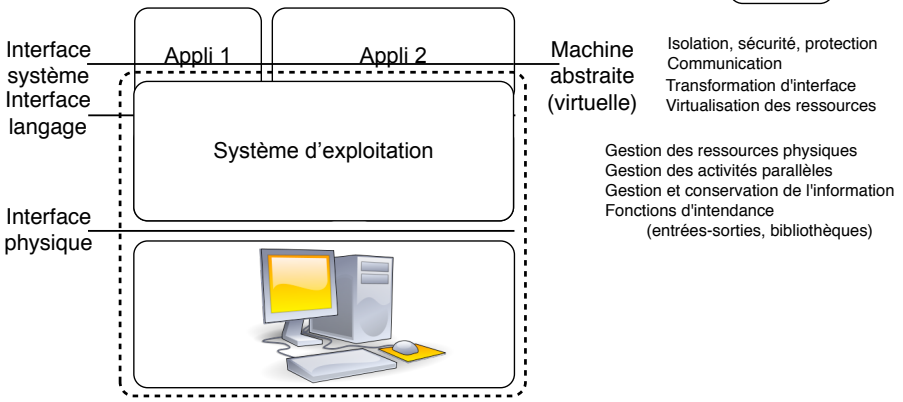
"An operating system is a collection of things that don't fit into a language. There shouldn't be one."

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

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

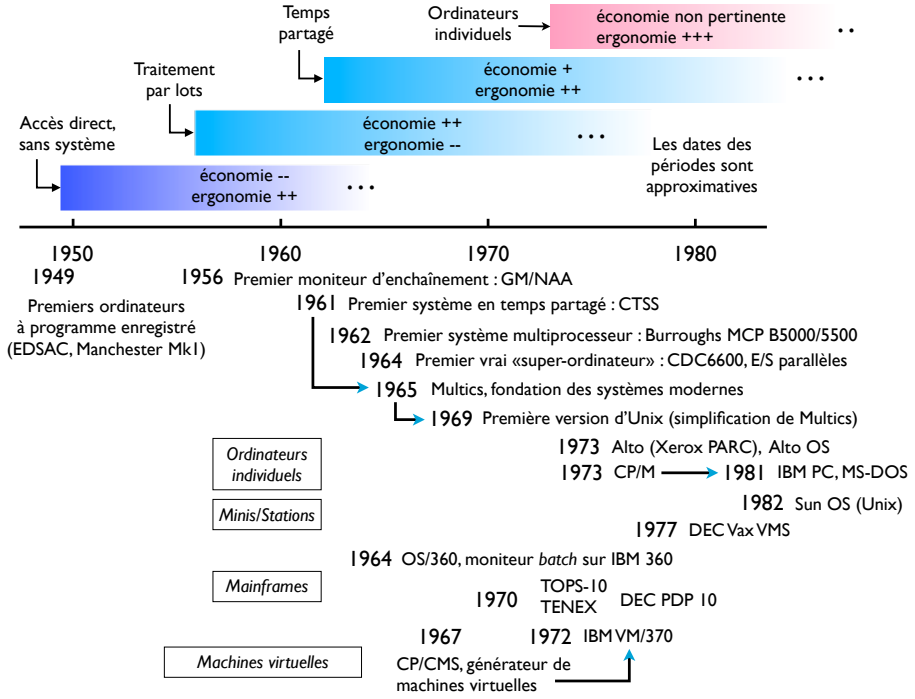
Qu'est-ce qu'un système d'exploitation ?

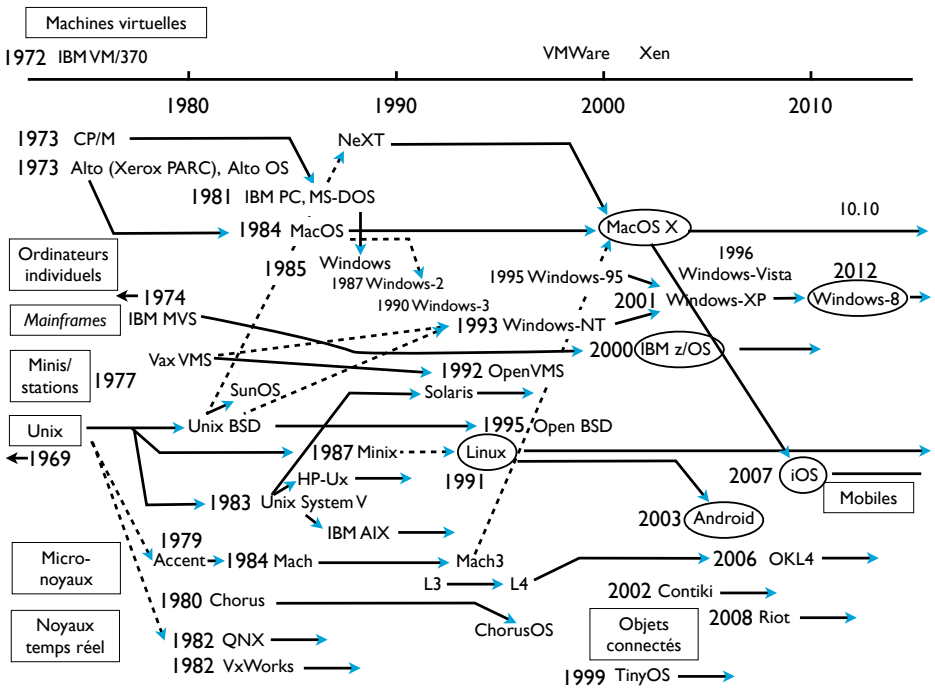




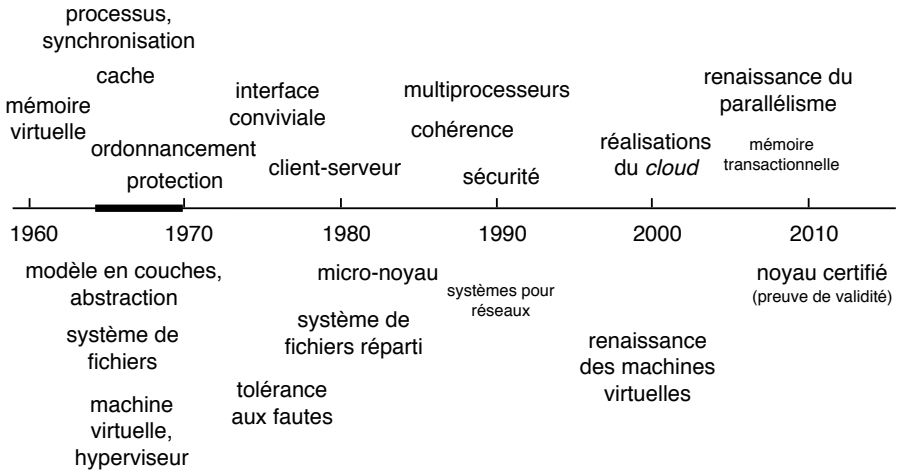
Qualités d'un système d'exploitation

- Discrétion
  - 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)

1956
Premier moniteur *batch*
Premier compilateur Fortran

- Bon confort, mais...
  - peu économique, vu le coût élevé du matériel
  - faible productivité, vu la pauvreté des outils

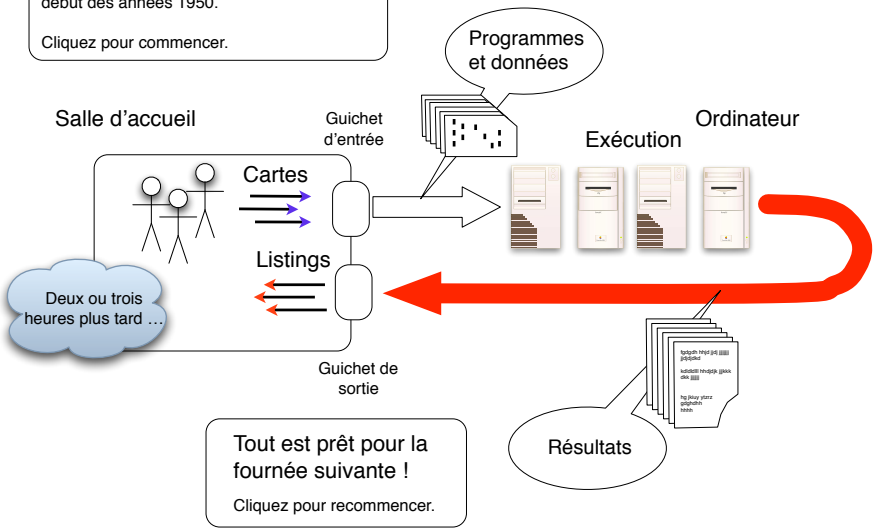
Systèmes de traitement par lots

- 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*)

Cette animation montre très schématiquement le fonctionnement d'un système d'exploitation à traitement par lots (*batch*) tel qu'il existait au début des années 1950.

Cliquez pour commencer.



Tout est prêt pour la fournée suivante !

Cliquez pour recommencer.

Un nouveau modèle d'exploitation

- L'utilisateur éloigné de la machine…
  - perte de l'interactivité
  - longs délais d'attente
  - faible tolérance aux erreurs
- De nouveaux métiers
  - atelier de préparation des travaux
    - suite des ateliers de mécanographie
    - «perfo - vérif»
  - gestion et surveillance de l'exécution
    - opérateur - pupitreur

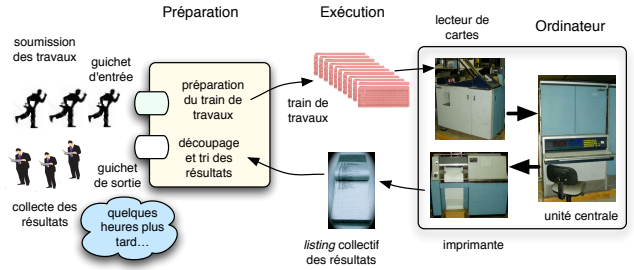
© FEB - Jean Bellec

Courtesy Computer History Museum

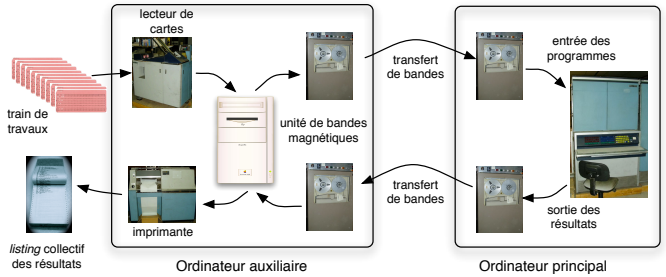
Avancées architecturales

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



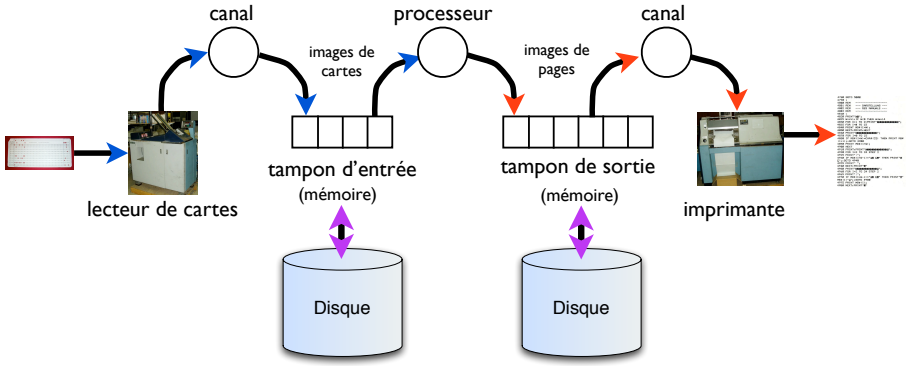
L'ordinateur principal et l'ordinateur auxiliaire fonctionnent en parallèle et peuvent être en des lieux différents.

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

Un système *batch* typique : OS/360 (1)

- Le défi
  - un système d'exploitation unique pour la gamme IBM/360
  - quelques compromis pour les petites configurations
- Les réalisations
  - PCP : un travail (*job*) à la fois
  - MFT : multiprogrammé, nombre fixe de tâches (1964)
  - MVT : multiprogrammé, nombre variable de tâches (1967)
    - exploitation quasi-optimale du matériel…
    - … mais peu de considération pour l'utilisateur
- Les extensions
  - TSO : *Time sharing option* (1971)
    - accès interactif, soumission et mise au point des travaux à distance
  - OS/VS1, VS/2 : mémoire virtuelle (1972)

Un système *batch* typique : OS/360 (2)

- Du côté de l'utilisateur
  - l'interface : JCL (*Job Control Language*)
    - indications pour le traitement (priorité, temps max, etc.)
    - définition des programmes et données
  - un langage peu convivial
    - une syntaxe rigide (liée au format des cartes)
    - une définition de données complexe (pré-bases de données)
- Du côté du génie logiciel
  - un système complexe (le plus gros logiciel de l'époque)
  - sans dessein architectural
    - (Fred Brooks : "*a multi-million dollar mistake*")
  - programmé en assembleur (initialement)
  - difficile à maintenir
    - ("*1000 bugs per release*")

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

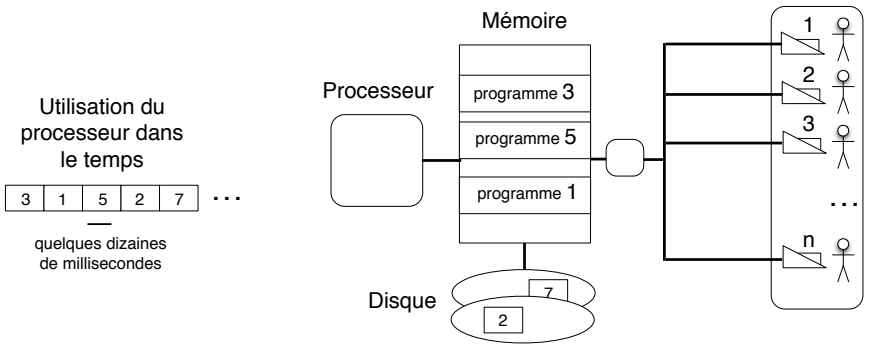
CTSS (*Compatible Time-Sharing System*)

- Un système novateur
  - premier système en temps partagé (compatible avec *batch*)
  - ordonnancement multi-niveaux
  - communication entre utilisateurs
  - langage de commande

Fernando Corbató
Jason Dorfman, MIT CSAIL

- 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
- Les retombées
  - démonstration de la viabilité du temps partagé
  - influence directe : Multics (MIT), CP/CMS (IBM)

Un système en 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)

- Une période charnière : 1965-70
  - 1967 : *First ACM Symposium on Operating Systems Principles*
- Un concept clé : la virtualisation

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

segment

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

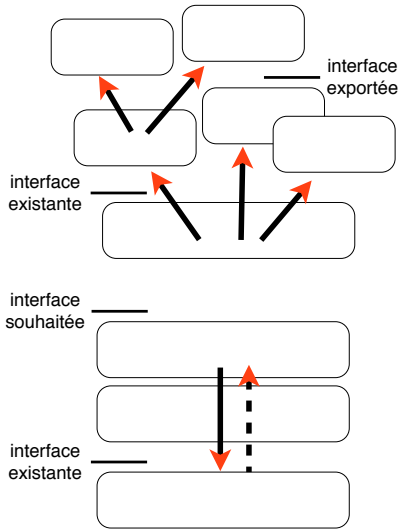
- Des expériences fondatrices
  - THE, Multics, CP67

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

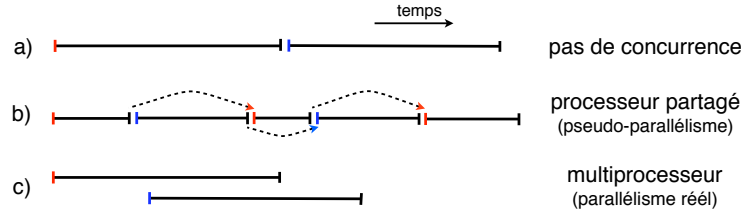
- Descendante (raffinement)
  - Un outil de conception
  - De la spécification à la réalisation
  - Hiérarchie de machines abstraites



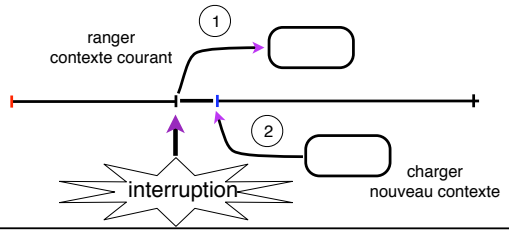
Programmes concurrents

- Les origines

  La multiprogrammation permet l'exécution concurrente de plusieurs programmes : divers schémas possibles



- Un mécanisme

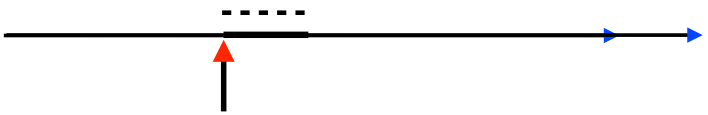


Dangers des interruptions (1)

«C'était une grande invention, mais aussi une boîte de Pandore.»

Edsger Dijkstra

- Effet d'une interruption (en l'absence de précautions)



insérer une séquence d'instructions «n'importe où» dans le corps d'un programme

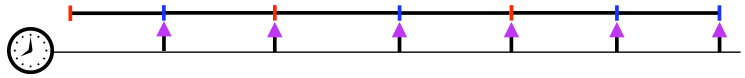
l'exécution du programme devient indéterministe (non reproductible)

il devient difficile (voire impossible) de raisonner sur l'effet du programme

Danger des interruptions (2)

- Exécution pseudo-parallèle

les changements de programme sont déclenchés par des interruptions d'horloge



exemple : deux opérations de dépôt sur un même compte bancaire

activité 1	activité 2
1.1: courant$_1$ = lire_compte (867A)	2.1: courant$_2$ = lire_compte (867A)
1.2: nouveau$_1$ = courant$_1$ + 1000	2.2: nouveau$_2$ = courant$_2$ + 3000
1.3: ecrire_compte (867A, nouveau$_1$)	2.3: ecrire_compte (867A, nouveau$_2$)

un scénario d'exécution : 1.1 ; 1.2 ; 2.1 ; 2.2 ; 2.3 ; 1.3

résultat net ?

le compte est crédité de 1 000 € au lieu de 4 000 € !

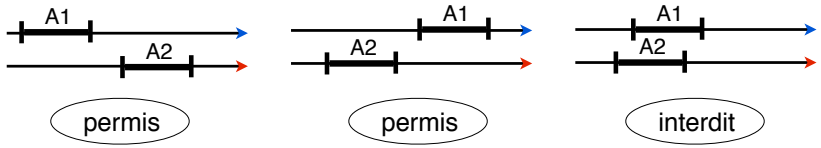
Un problème de synchronisation

- Comment éviter la perte de mise à jour ?

un remède : imposer une exécution séquentielle des deux opérations de dépôt (A1 et A2)

soit A1 puis A2, soit A2 puis A1 : tout entrelacement est interdit

autrement dit : une seule des activités (au plus) peut exécuter à un instant donné l'opération en cause (dite section critique)

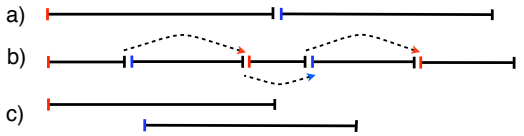


Cette condition s'appelle l'exclusion mutuelle ; c'est le premier problème de synchronisation explicitement formulé (Dijkstra, 1965)

La notion de processus

- Un pas vers l'abstraction…

qu'y a-t-il de commun entre les exécutions suivantes ?



Edsger W. Dijkstra
(1930 - 2002)
crédit : University of Texas at Austin

réponse : deux activités considérées comme séquences d'actions, indépendamment des conditions de leur exécution.

on arrive ainsi à la notion de **processus** (séquentiel), programme en cours d'exécution, indépendamment de ses ressources

chaque processus a un processeur virtuel (éventuellement à vitesse nulle)

avantage : séparer les aspects logiques de l'allocation de ressources

Synchronisation des processus

- Objectif
  - imposer des contraintes logiques entre processus pour régler la compétition ou la coopération
- Moyens
  - bloquer (faire attendre) un processus jusqu'à réalisation d'une condition
- Outils
  - instructions spéciales
  - verrous
  - sémaphores
  - moniteurs
  - messages
- Exemple

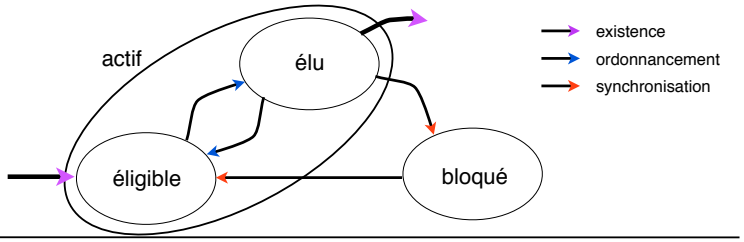
exclusion mutuelle

processus p1	processus p2
…	…
verrouiller (v)	verrouiller (v)
section critique	section critique
déverrouiller (v)	déverrouiller (v)
…	…

Réaliser les processus : le noyau

Deux fonctions

- Allouer le(s) processeur(s)
  - Choisir une politique d'ordonnancement
  - Exemple : le tourniquet (*round robin*), plus court d'abord, etc.
- Fournir une interface
  - Création, destruction, examen des processus
  - Réalisation des primitives de synchronisation

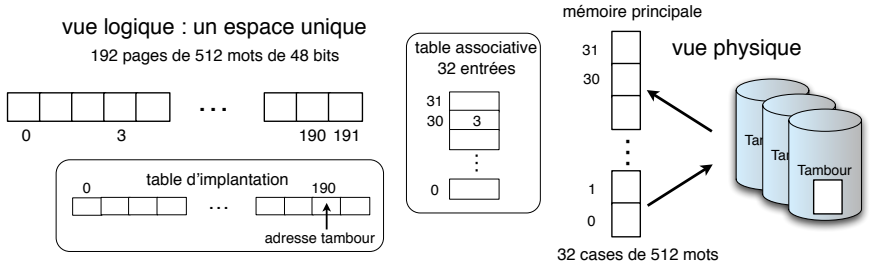


Virtualiser la mémoire (1)

- Deux problèmes à résoudre
  - Définir la structure d'une «mémoire virtuelle» (telle que vue par l'utilisateur)
  - Réaliser la correspondance entre mémoire virtuelle et mémoire physique
- Structure de la mémoire virtuelle
  - Mémoire linéaire (peut être plus grande que la mémoire physique)
  - Mémoire segmentée (ex : Burroughs MCP, Multics)
    - ensemble de segments de taille variable
    - schéma idéal, mais peu utilisé à l'état «pur»
- Mise en œuvre de la mémoire virtuelle
  - Pagination à la demande

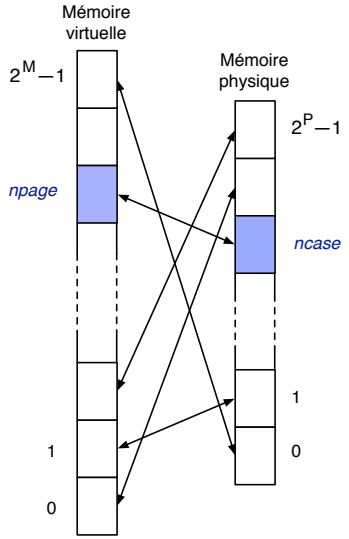
Virtualiser la mémoire (2)

- Un système précurseur…
  - L'Atlas (Manchester, 1961)
  - Concept de «mémoire à un niveau» (*one level store*)
  - Un espace unique, réalisé par mémoire principale et secondaire

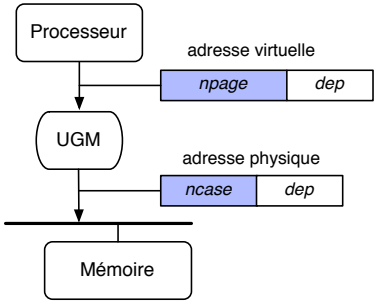


- … mais des problèmes inattendus
  - Dégradation brutale et inexpliquée des performances (écroulement)
  - L'explication viendra… en 1968

Virtualiser la mémoire (3)

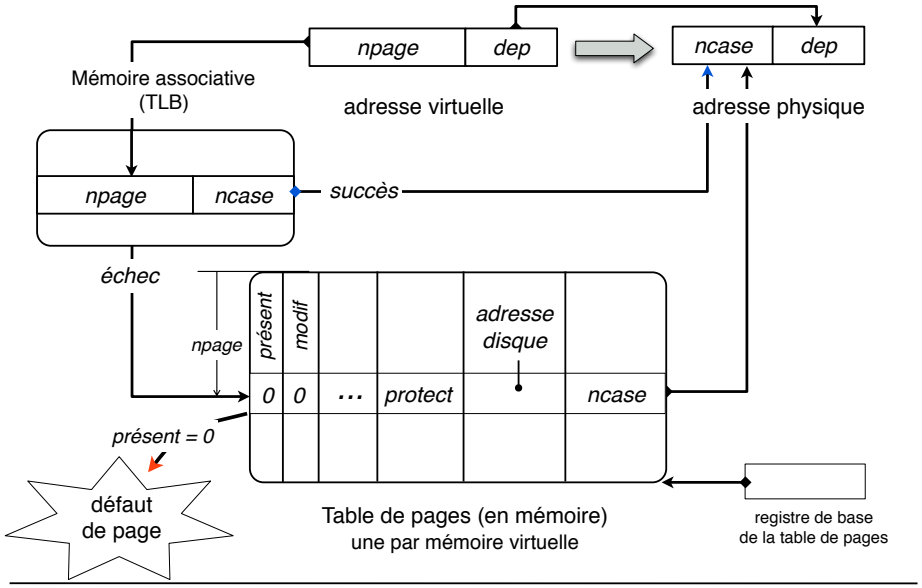


(a) Correspondance pages-cases



(b) Traduction des adresses

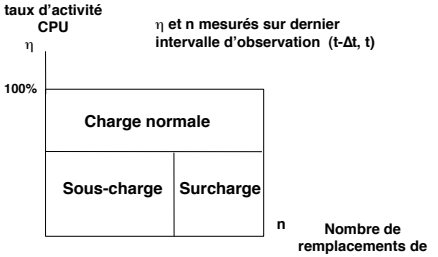
Pagination à la demande



Un piège de la mémoire virtuelle

- L'écroulement (*thrashing*)
  - Lorsque la charge augmente, le système «s'écroule»
    - explosion du taux d'activité du disque et du temps de réponse
    - baisse d'activité du processeur
- Explication schématique
  - Comportement des programmes
    - tout programme a besoin d'une taille minimale pour fonctionner dans de bonnes conditions
    - au-dessous de cette taille, le nombre de défauts de page augmente très vite
  - Coût élevé du défaut de page
    - T temps moyen de traitement d'un défaut de page
    - t temps moyen d'exécution d'une instruction
    - T / t de l'ordre de 10^3 à 10^4

Comprendre et éviter l'écroulement

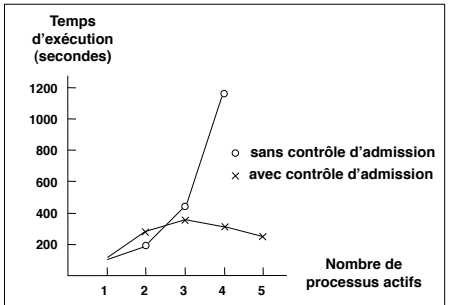


a) modèle simple de comportement du système

```
every Δt do
    if (overload)
        move one process from ready set to waiting set
    else
        if (underload and (waiting set ≠ ∅))
                admit one waiting process to ready set
```

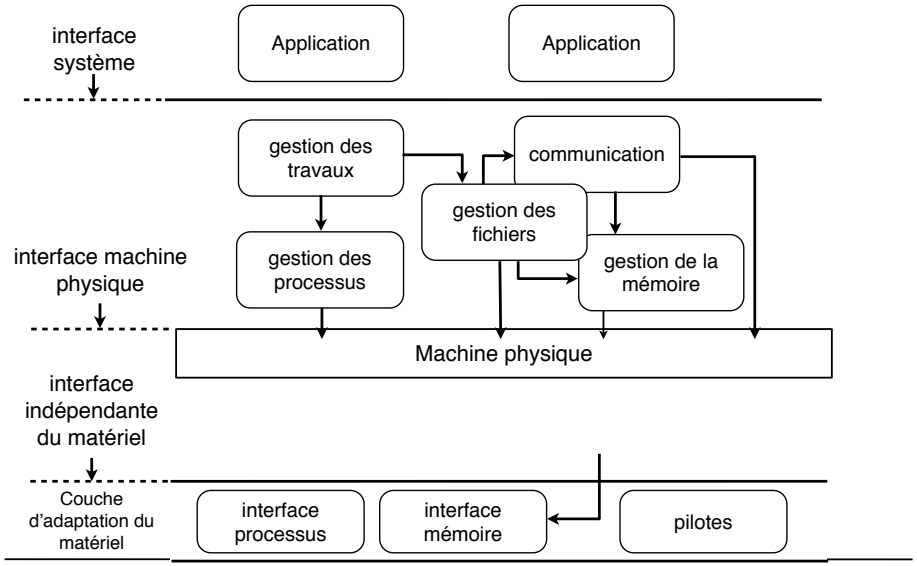
Prévention de l'écroulement : expériences IBM M44/44X (1968)

B. Brawn, F. Gustavson. Program behavior in a paging environment. *Proc. AFIPS FJCC*, pp. 1019-1032 (1968)

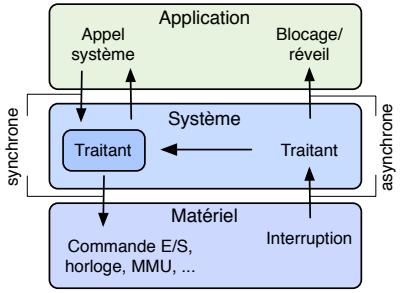


b) effet de la limitation de charge par contrôle d'admission

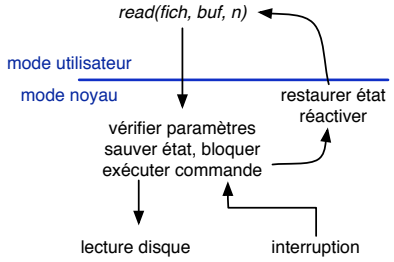
Structure d'un système d'exploitation (1)



Communication interne
dans un système d'exploitation



(a) Flot de contrôle

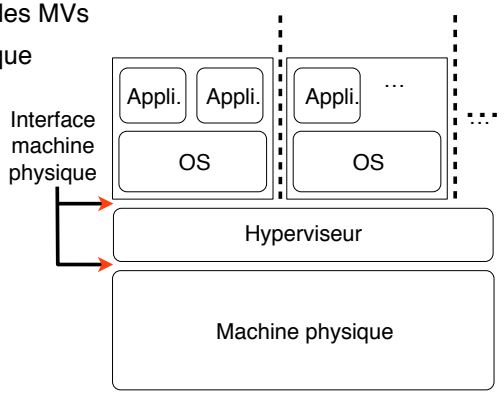


(b) Appel système

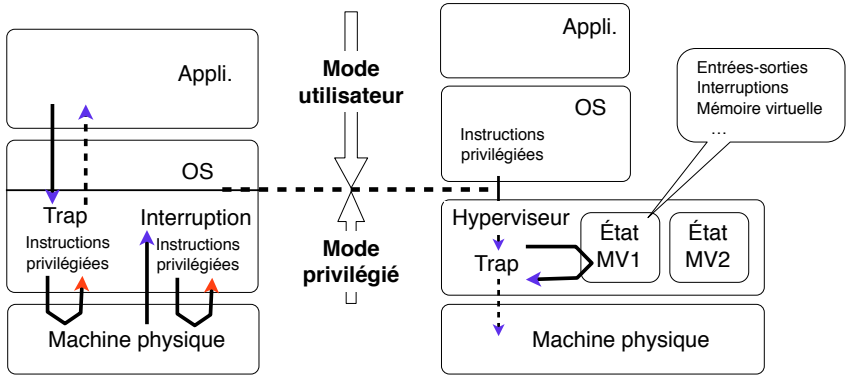
Causes d'interruption	**Appel système**	**Déroutement**
Horloge	Opération sur fichier	Opération impossible
Entrée-sortie	Entrée-sortie	Débordement
Organe externe	Opération sur processus	Erreur d'adressage
Incident		

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
- Première réalisation
  - CP67 (IBM 360/67)
  - Utilisation : CP/CMS
  - Base pour VM/370



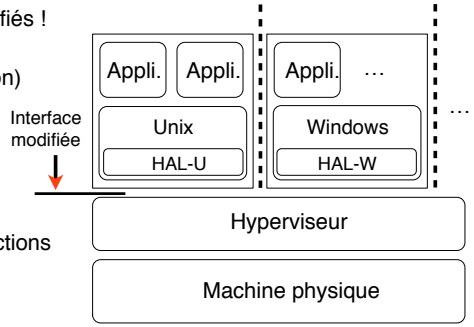
Machines virtuelles : comment çà marche



- Un problème …
  - Sur les machines actuelles (IA-32, etc.), l'effet de certaines instructions dépend du mode courant (privilégié 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

Sécurité et protection (1)

- Un exemple de séparation politique-mécanismes
  - Sécurité : définition de politiques visant à fixer des règles d'accès aux ressources
  - Protection : ensemble de mécanismes permettant de mettre en œuvre des politiques de sécurité
- Sécurité dans un système d'exploitation
  - Confidentialité (qui peut accéder aux données, pour faire quoi)
  - Intégrité (pas de modification non désirée)
  - Accès aux services
  - Authentification (prouver que l'on est celui qu'on prétend être)
- Protection : avancées 1968-70
  - Agents et objets
  - Droits, domaines, capacités

Sécurité et protection (2)

- Un problème (toujours) non résolu…
- Notion de domaine
  - un espace contenant des données et des points d'accès à des services
  - l'accès à ce contenu est protégé
    - points de passage obligés («guichets»)
    - contrôle des droits d'accès
  - Exemples
    - les modes maître-esclave
    - les anneaux de Multics
    - les systèmes à capacités (clé-serrure)
- Mais le système n'est pas invulnérable…

Failles de protection (1)

"*You can't trust code that you did not totally create yourself (especially code from companies that employ people like me)*"

«Vous ne pouvez pas vous fier à du code que vous n'avez pas entièrement créé vous-même (et particulièrement s'il vient d'entreprises qui emploient des gens comme moi)»

Ken Thompson (1984)

- Un point faible : l'accès au code objet
  - Développeurs, distributeurs, installateurs de mises à jour…
  - Insertion de code malveillant dans les compilateurs…
  - Une faille dans le code objet est très difficile à détecter
- Des points d'entrée non prévus
  - Cheval de Troie
  - Porte dérobée
- Exemple : attaques contre Multics

Failles de protection (2)

- Exploiter les situations d'erreur
  - la réaction aux défaillances est souvent moins sécurisée que le code «normal»
  - exemples
    - débordement de tampon
      - introduire du code exécutable dans une zone de données
    - détournement d'un mécanisme de pagination
      - découvrir rapidement un mot de passe dans le système Tenex
- Utiliser des canaux cachés
  - communication illégale entre domaines spécifiés comme isolés
    - peut altérer la confidentialité ou le droit d'accès
- Exploiter la vulnérabilité des réseaux
  - exemple : *man in the middle*

L'époque moderne

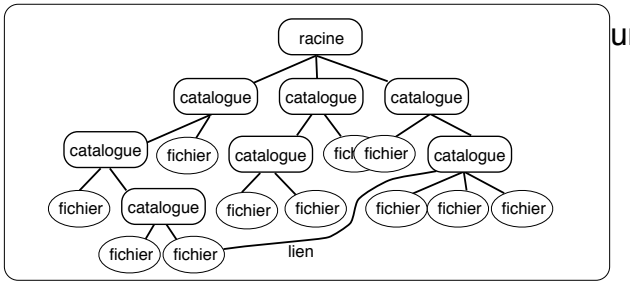
(après l'avènement de l'approche scientifique)

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



ures

Vie de Multics

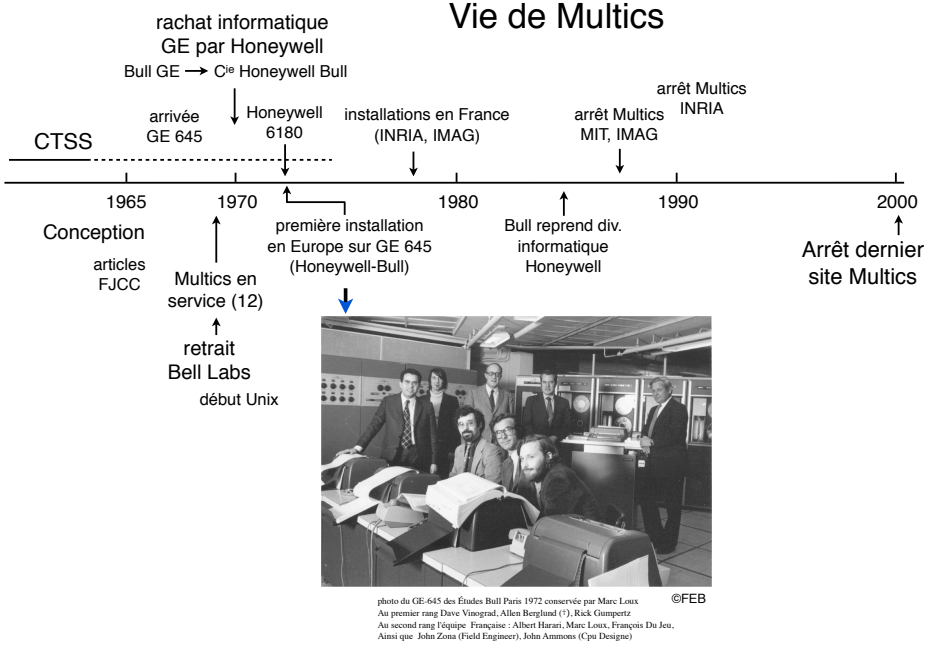


photo du GE-645 des Études Bull Paris 1972 conservée par Marc Loux
Au premier rang Dave Vinograd, Allen Bergluud (†), Rick Gumpertz
Au second rang l'équipe Française : Albert Harari, Marc Loux, François Du Jeu,
Ainsi que John Zona (Field Engineer), John Ammons (Cpu Designe)

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

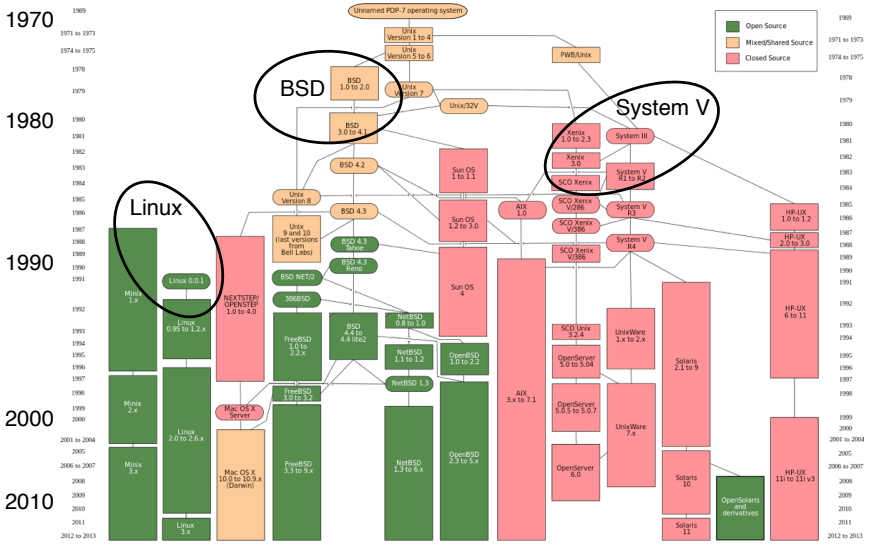
1979-80

Premières versions industrielles

1988

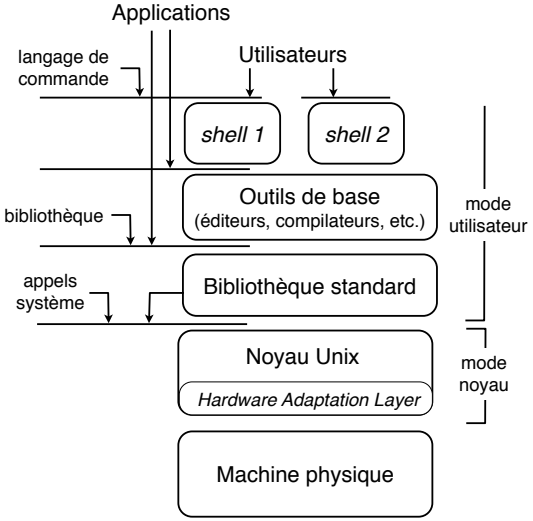
Première version de Posix

La famille Unix



source : Wikipedia, Levenez Unix History Diagram

Unix : architecture d'ensemble



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

Unix : principes de base

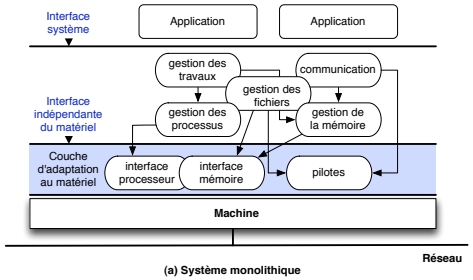
- Entité de base : le processus
  - mémoire virtuelle linéaire + flots d'exécution (*threads)*
  - segments «mappés» dans la mémoire virtuelle
  - mémoire virtuelle partagée
- Fichiers
  - système hiérarchique à la Multics + protection simple
  - descripteur (*inode*) + tables d'implantation multiniveaux
  - 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)

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

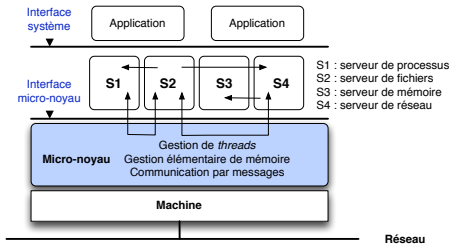
Domaine actif dans les années 1980 à 1995
En retrait depuis, mais renouveau récent (systèmes embarqués)

Système sur micro-noyau



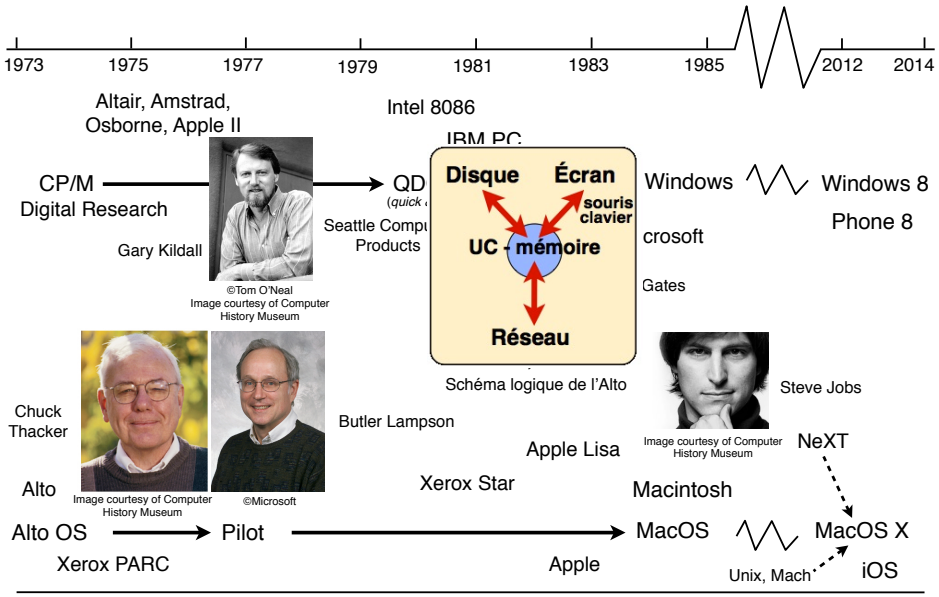
(a) Système monolithique

Exemples
- Windows NT
- Mac OS X (Darwin sur Mach 3)



(b) Système à micro-noyau

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

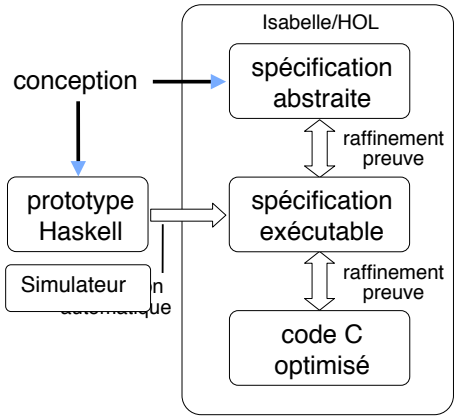
Virtualisation dans les systèmes embarqués

- 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 (IIES'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 "microviseur"
  - 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