Systèmes et Réseaux – MIAGE 2
Systèmes d'Exploitation

Introduction

Sara Bouchenak

Sara.Bouchenak@imag.fr
http://sardes.inrialpes.fr/~bouchena/teaching

Systèmes et Réseaux

- Objectifs
  - Présentation des principaux concepts des systèmes d'exploitation et des réseaux
  - Point de vue de l'utilisateur
- Organisation
  - 1ère partie : Systèmes
    - Chargé de cours : Sara Bouchenak
    - 9h TP
  - 2ème partie : Réseaux
    - Chargé de cours : Alain Cartade
    - TP
  - 3ème partie : Projet
    - 18h

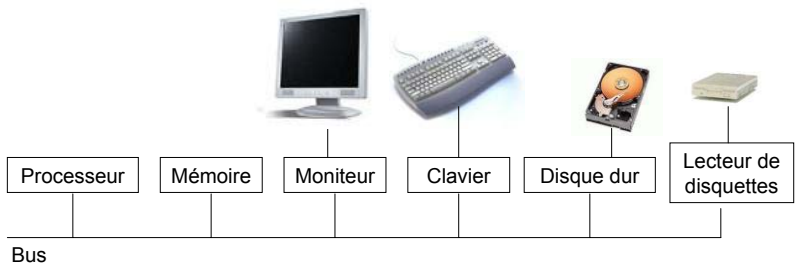
Organisation du cours de systèmes

Date	Cours	TP	Projet
23/09/2005	Introduction		
07/10/2005	Processus		
07/10/2005	Gestion des processus		
14/10/2005	Fichiers	Processus	
21/10/2005	Mémoire	Interruptions	
vacances			
04/11/2005	Etude de cas	Fichiers	
Cours réseaux			
19/12/2005			Bibliothèque répartie
20/12/2005			Bibliothèque répartie
21/12/2005			Bibliothèque répartie

Plan

1. **Introduction aux systèmes d'exploitation**
   - Qu'est-ce qu'un système d'exploitation
   - Historique
   - Concepts de base
   - Interfaces d'un système d'exploitation
2. Processus
3. Gestion des processus
4. Fichiers
5. Mémoire
6. Etude de cas

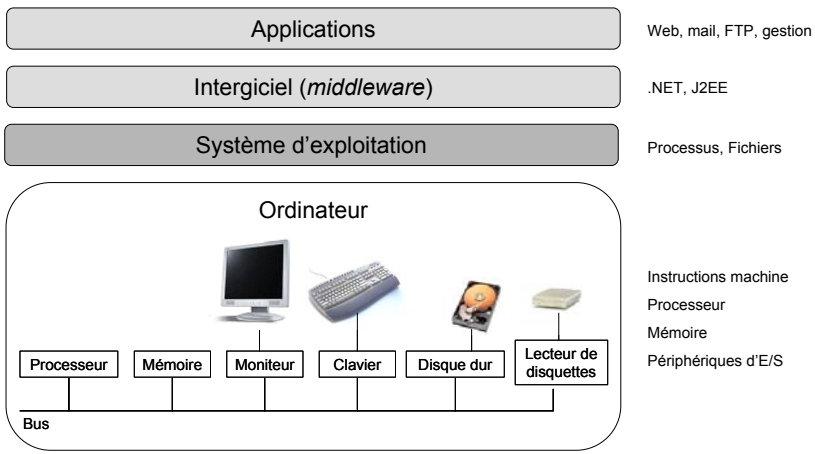
Structure matérielle d'un ordinateur (1)



Structure matérielle d'un ordinateur (2)

- Processeur
  - "Cerveau" de l'ordinateur
  - Jeu d'instructions machine (ex. charger un mot mémoire dans un registre, stocker le contenu d'un registre dans la mémoire, etc.)
  - Ensemble de registres (ex. compteur ordinal)
  - Exécute les instructions machine
- Mémoire
  - Espace contenant les programmes à exécuter et leurs données
- Périphériques d'Entrées/Sorties
  - Disque dur, clavier, écran
  - Contrôleur de périphérique d'E/S

Système d'exploitation



Rôle d'un système d'exploitation (1)

- Interface de plus haut niveau
  - Le système fournit à ses utilisateurs une interface plus commode à utiliser que le matériel
    - Dissimule les détails de mise en œuvre
    - Dissimule les limitations physiques (nombre de processeurs, taille mémoire) et le partage des ressources entre plusieurs utilisateurs
- Gestion des ressources
  - Le système gère les ressources matérielles : mémoire, processeur
  - Cette gestion comprend l'allocation, le partage, la protection

Rôle d'un système d'exploitation (2)

- Virtualisation de l'ordinateur sous-jacent
  - Ordinateur :
    - ensemble de ressources physiques (réelles)
    - ex. processeur, disque dur
  - Système d'exploitation :
    - facilite l'utilisation des ressources physiques
    - permet le partage de ressources entre plusieurs applications à la fois

Composant au niveau ordinateur	Concept virtualisé au niveau système
Processeur	Processus
Disque	Fichier

Plan

1. **Introduction aux systèmes d'exploitation**
   - *Qu'est-ce qu'un système d'exploitation*
   - Historique
   - Concepts de base
   - Interfaces d'un système d'exploitation
2. Processus
3. Gestion des processus
4. Fichiers
5. Mémoire
6. Etude de cas

Historique des systèmes d'exploitation

- Première génération (1945-1955) : tubes à vide et tableaux d'interrupteurs
  - Moteurs de calcul à relais mécaniques
    - Très lents (temps de cycles mesurés en secondes)
  - Tubes à vide
    - Relais mécaniques remplacés par des tubes à vide
    - Machines énormes remplissant des pièces entières (dizaines de milliers de tubes à vide)

Historique : 1ère génération

- La même équipe concevait, construisait, programmait, administrait et maintenait la machine
- Tout programme était conçu en langage machine (les autres langages n'existaient pas)
- Un programme était conçu en basculant des tableaux d'interrupteurs pour contrôler les fonctions de base de la machine (on programmait la machine avec un programme)

Historique : 1ère génération (suite)

- Protocole classique d'utilisation de la machine
  - Le programmeur demande une réservation de la machine pour une certaine durée
  - Il insère son programme dans la machine en manipulant le tableau d'interrupteurs
  - Dans les heures qui suivent, il prie pour qu'aucun des quelques 20.000 tubes ne grille pendant l'exécution
- Protocole amélioré avec les cartes perforées
  - Un programme est écrit sur des cartes
  - Les cartes sont lues par la machine au lieu d'utiliser des tableaux d'interrupteurs pour 'programmer la machine'

Historique : 1ère génération (fin)

- Types de programmes
  - Simples calculs numériques
- Et le système d'exploitation ?
  - Il n'existait pas encore

Historique des systèmes d'exploitation

- Deuxième génération (1955-1965) : transistors et systèmes par lots
  - Transistors
    - Ordinateurs plus fiables
    - Séparation nette entre concepteurs, constructeurs, programmeurs, opérateurs et personnel de maintenance
  - Les programmes étaient écrits en FORTRAN puis codés sur des cartes perforée

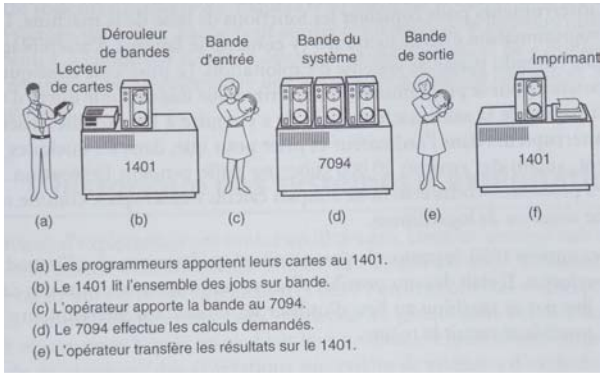
Historique : 2ème génération

- Protocole d'utilisation de la machine
  - Le programmeur apporte son paquet de cartes dans la salle de soumission des *jobs* (tâches, programmes à exécuter)
  - L'opérateur fait lire et exécuter les cartes par la machine
  - L'opérateur récupère la trace d'exécution sur une imprimante et la stocke dans la salle des résultats pour que le programmeur la récupère
  - L'opérateur prend ensuite un autre paquet de cartes soumis et répète le processus précédent
  - De plus, si le compilateur FORTRAN est nécessaire, l'opérateur doit également le charger dans la machine

Historique : 2ème génération (suite)

- Protocole lent
  - Un opérateur humain traite séquentiellement un job et gère la soumission :
    - des entrée (lecture de cartes perforée)
    - des sorties (résultat sur imprimante) une fois le calcul terminé
- Amélioration du protocole
  - Découpage du traitement d'un job en plusieurs étapes
    - Lecture des entrées, calcul, production des sorties
    - Entrées/sorties d'un job traitées par une machine peu onéreuse et performante pour les opération d'E/S
    - Calcul effectué par une machine performante (calculateur)
  - Batch : traitement par lots

Historique : 2ème génération (illustration)



[*A. Tanenbaum. Systèmes d'exploitation – 2ème édition. Pearson Education, 2003*]

Historique : 2ème génération (fin)

- Types de programmes
  - Calculs scientifiques et d'ingénierie (ex. résolution d'équations aux dérivées partielles)
  - Programmés en FORTRAN ou en langage d'assembleur
- Système d'exploitation
  - FMS (*Fortran Monitor System*)
  - IBYS (1er système d'exploitation de l'IBM 7094)

Historique des systèmes d'exploitation

- Troisième génération (1965-1980)
  - Circuits intégrés
    - Avantage au plan prix/performance
  - Une seule gamme de produits
    - Début 60, des machines spécialisées dans les E/S (IBM 1401) et d'autres spécialisées dans le calcul (IBM 7094)
    - Puis une série de machines (taille entre l'IBM 1401 et l'IBM 7094) avec la même architecture matérielle et le même jeu d'instructions, ex. system/360
    - Un même programme peut en principe tourner sur toutes ces machines

Historique : 3ème génération

- Système d'exploitation
  - OS/360, devait tourner sur tous les modèles (du plus petit au plus grand)
- Nouvelles techniques dans les systèmes d'exploitation
  - Multiprogrammation
    - Plusieurs programmes pouvant s'exécuter en parallèle sur une même machine
    - Mémoire partagée
    - Mécanismes de protection
  - Spool (*Simultaneous Peripheral Operation Onl Line*)
    - Lectures et écritures simultanées sur le disque
    - IBM 1401 n'est plus nécessaire
    - Manipulations de bandes ont disparu

Historique : 3ème génération (fin)

- Exemples de systèmes d'exploitation
  - MULTICS, du MIT, Bell Labs et General Electric
    - Système capable de supporter des centaines d'utilisateurs simultanés
  - Systèmes pour mini-ordinateurs, DEC PDP
    - PDP-1 … PDP-11
    - Peu d'espace mémoire (4 K-mots de 18 bits)
    - Performants pour des tâches non scientifiques
  - UNIX
    - Ken Thompson de Bell Labs, qui a travaillé sur MULTICS, a écrit une version allégée de MULTICS pour un PDP-7
    - Norme POSIX
  - Linux (Linus Torvalds)
    - Proche d'Unix
    - Version commerciale

Historique des systèmes d'exploitation

- Quatrième génération (1980-aujourd'hui) : ordinateurs personnels
  - Circuits intégrés à haute densité
    - Puces contenant des milliers de transistors sur 1mm^2 de silicium
    - Micro-ordinateurs, très peu onéreux comparés aux mini-ordinateurs de type PDP-11

Historique : 4ème génération

- Exemples de systèmes d'exploitation
  - CP/M (*Control Program for Microcomputers*)
    - En 1974, CP/M est conçu pour Intel par leur consultant Gary Kildall pour le processeur Intel 8080
    - Comprend un contrôleur pour le tout récent lecteur de disquettes 8 pouces
    - Intel pense que les µ-ordinateurs équipés de disque ont peu de chance de prospérer, cède à G. Kildall les droits du CP/M
    - En 1977, Digital Research est fondé, vend des CP/M écrits pour des processeurs 8080, Zilog Z80 et autres

Historique : 4ème génération (suite)

- MS-DOS (MicroSoft-Disk Operating System)
  - En 1980, IBM propose l'IBM PC
  - IBM demande à Bille Gates s'il connaît un système d'exploitation pouvant fonctionner sur l'IBM PC
  - B. Gates conseille à IBM de contacter Digital Research, la plus grande entreprise eu monde dans le domaine des systèmes d'exploitation, mais refus de la part de Digital R.
  - IBM a finalement demandé à B. Gates de lui fournir un système d'exploitation
  - B. Gates a acheté DOS à un petit constructeur de Seattle et après quelques modifications, il a été baptisé (MS-DOS)
  - B. Gates décide de vendre MS-DOS aux constructeurs pour que le système d'exploitation soit directement fourni aux utilisateurs, plutôt que de le vendre directement aux utilisateurs comme l'a fait G. Kildall

Historique : 4ème génération (suite)

- Macintosh d'Apple
  - Steve Jobs (co-inventeur d'Apple) visite le centre de recherche de Xerox par cet découvre l'invention du concept d'IHM graphique
  - S. Jobs construit un Apple avec une IHM graphique
  - Lisa, trop cher, échec commercial
  - Macintosh, beaucoup moins cher, très convivial
  - Destiné à des utilisateurs qui ne connaissaient rien aux ordinateurs et qui n'aveint aucunement l'intention d'en apprendre quoi que ce soit

Historique : 4ème génération (fin)

- Windows de Microsoft
  - Successeur de MS-DOS
  - Influencé par le succès de Macintosh
  - Intègre une IHM graphique
  - Autres successeurs : Windows 95/98/NT/2000/XP
- UNIX
  - Intégration de l'IHM graphique
  - Système de fenêtres appelé X-Window (MIT)
  - IHM complète telle que Motif au-dessus de X-Window

Historique : Résumé

- 1ère génération (1945-1955)
  - Tubes à vides, tableaux d'interrupteurs, cartes perforées
  - Préhistoire des systèmes d'exploitation
- 2ème génération (1955-1965)
  - Transistors, traitements par lots
  - Systèmes : FMS, IBYS
- 3ème génération (1965-1980)
  - Circuits intégrés (mini-ordinateurs, faible rapport qualité/prix)
  - Nouveaux concepts : multiprogrammation, spool
  - Systèmes : OS/360, MULTICS, systèmes pour PDP UNIX, Linux
- 4ème génération (1980-aujourd'hui)
  - Circuits intégrés à haute densité (μ-ordinateurs, peu chers)
  - Nouveau concept : IHM graphique
  - Systèmes : CP/M, MS-DOS, Macintosh, Windows

Plan

1. **Introduction aux systèmes d'exploitation**
   - *Qu'est-ce qu'un système d'exploitation*
   - *Historique*
   - Concepts de base
   - Interfaces d'un système d'exploitation
2. Processus
3. Gestion des processus
4. Fichiers
5. Mémoire
6. Etude de cas

Concepts de base des systèmes d'exploitation

- Processus
- Interblocage
- Gestion de la mémoire
- Entrées/sorties
- Fichiers

Processus

- Processus = programme en cours d'exécution
- Chaque processus est pourvu d'un espace d'adressage en mémoire et d'un ensemble de registres
- Cet espace contient le programme exécutable, les données et la pile d'exécution
- Ex. de registres : compteur ordinal, sommet de pile

Interblocage entre processus

- Interblocage = deux ou plusieurs processus se retrouvent dans une situation dont ils ne peuvent plus s'échapper
- Exemple d'interblocage entre deux processus
  - Ordinateur équipé d'un lecteur de disquette et d'un graveur de CD-ROM
  - Deux processus distincts désirent graver un CD-ROM à partir d'une disquette
  - Le processus 1 demande d'abord l'accès au lecteur de disquette (il lui sera accordé)
  - Entre temps, le processus 2 réquisitionne (avec succès) le graveur
  - Puis, le 1er processus demande l'accès au graveur qui lui sera refusé
  - Et le 2ème processus demande l'accès au lecteur qui lui sera refusé
  - Les deux processus sont bloqués en attente d'une ressource qu'ils n'obtiendront jamais puisqu'elle ne sera jamais libérée

Gestion de la mémoire (1)

- Tout ordinateur possède une mémoire principale qui contient le programme en cours d'exécution
- Monoprogrammation (cas simple)
  - Seul un programme à la fois peut se trouver en mémoire
  - Pour exécuter un second programme, on doit d'abord décharger le 1er programme de la mémoire puis charger le second

Gestion de la mémoire (2)

- Multiprogrammation
  - Plusieurs programmes peuvent cohabiter en même temps en mémoire
  - Mécanisme de protection qui empêche deux programmes d'interférer entre eux
  - Mécanisme de gestion de l'espace d'adressage
    - Un processus possède un espace d'adressage supérieur à celui de la mémoire principale.
      Comment fait-il pour s'exécuter ?
      - Dans les premiers ordinateurs, un tel processus échouait
      - Puis technique de mémoire virtuelle qui permet au système d'exploitation de garder une partie de l'espace d'adressage d'un processus en mémoire principale et une partie sur le disque, et d'effectuer les allers-retours nécessaires entre les deux

Entrées/sorties (1)

- Les ordinateurs sont munis de périphériques
  - acquisition de données en entrée
    - Clavier, souris
  - production de données en sorties
    - Écran, imprimante
- Le système d'exploitation possède un sous-système d'entrées/sorties responsable de la gestion des périphériques

Entrées/sorties (2)

- Le sous-système d'entrées/sorties dans un système d'exploitation contient :
  - Une partie générique, indépendante du périphérique et s'appliquant donc à plusieurs périphériques
  - Une partie spécifique à des périphériques particuliers
    - Ex. le pilote (*driver*) d'une imprimante HP Laser Jet

Fichiers (1)

- Les fichiers sont un mécanisme d'abstraction (ou de virtualisation) du disque
- Les fichiers permettent d'écrire/lire des données sur un disque
- L'utilisateur ne voit pas :
  - comment sont stockées les données
  - où sont stockées les données
  - comment les disques fonctionnent

Fichiers (2)

- Exemple d'abstraction fournie par les fichiers
  - Lecture d'un fichier vs. localisation d'un bloc sur un secteur sur le disque
- Système de gestion de fichiers (SGF) = sous-système d'un système d'exploitation
  - Ex. NFS (*Network File System*) dans UNIX

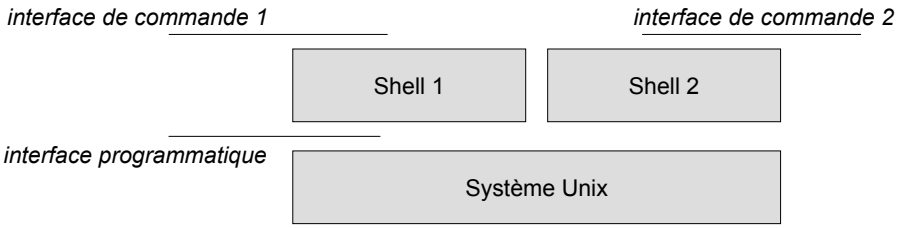
Plan

1. **Introduction aux systèmes d'exploitation**
   - *Qu'est-ce qu'un système d'exploitation*
   - *Historique*
   - *Concepts de base*
   - Interfaces d'un système d'exploitation
2. Processus
3. Gestion des processus
4. Fichiers
5. Mémoire
6. Etude de cas

Interfaces d'un système d'exploitation (1)

- L'interface d'un système d'exploitation permet aux utilisateurs du système de faire appel aux fonctions du système
- Un système d'exploitation présente en général deux interfaces
  - Une interface programmatique
  - Une interface de commande

Interfaces programmatique vs. interface de commande



Interfaces d'un système d'exploitation (2)

- Interface programmatique (API – *Application Programming Interface*)
  - Dédiée aux programmes s'exécutant au-dessus d'un système d'exploitation (ex. intergiciel, application)
  - Composée d'un ensemble de fonctions système avec leurs paramètres
- Interface de commande (shell)
  - Dédiée aux utilisateurs humains d'un système d'exploitation (ex. administrateur)
  - Composée d'un ensemble de commandes
    - Commandes textuelles : rm *.o
    - Commandes graphiques : déplacer l'icône d'un fichier dans une corbeille

Interface de commande dans Unix – Shell (1)

- Le shell est un programme qui interprète des commandes
- Entrée standard du shell : clavier
- Sortie standard du shell : écran
- Exemple de commande de shell
  - Commande *date* affiche la date courante à l'écran
    - % date

Interface de commande dans Unix – Shell (2)

- Entrées/sorties des commandes de shell
  - % date > fichier
  - Redirection de la sortie de commande dans un fichier
  - % sort < fichier_entree > fichier_sortie
  - Récupération de l'entrée de la commande d'un fichier et redirection de sa sortie dans un fichier
  - % cat fichier1 fichier2 | sort > /dev/lp
  - La sortie d'une commande est redirigée vers l'entrée d'une autre commande, le résultat final est envoyé à l'imprimante

API Unix pour la gestion des processus

Fonction	Description
pid = fork()	Crée un processus fils
pid = waitpid(pid, &stat, options)	Attend la terminaison d'un processus fils
s = execve(nom, argv, env)	Remplace l'image d'un processus
exit(statut)	Termine l'exécution d'un processus et renvoie le résultat

API Unix pour la gestion des fichiers

Fonction	Description
df = open(fich, mode, …)	Ouvre un fichier en lecture et/ou écriture
s = close(df)	Ferme un fichier ouvert
n = read(df, tampon, nbOctets)	Lit des données d'un fichier dans un tampon
n = write(df, tampon, nbOctets)	Ecrit des données d'un tampon dans un fichier
pos = lseek(df, offset, orig)	Déplace le pointeur du fichier
s = stat(nom, &buf)	Récupère l'information sur un fichier

API Unix pour la gestion des répertoires et autres

Fonction	Description
s = mkdir(nom, mode)	Crée un nouveau répertoire
s = rmdir(nom)	Supprime un répertoire vide
s = chdir(nom)	Change le répertoire de travail
s = chmod(nom, mode)	Change les droits d'accès à un fichier ou répertoire
s = kill(pid, signal)	Envoie un signal à un processus
sec = time(&sec)	Renvoie le temps écoulé depuis le 1er janvier 1970

Interfaces du système Unix

- Documentation en ligne avec la commande de manuel *man*
  - man 1 <nom de la commande> : documentation des commandes (option par défaut du *man*)
  - man 2 <nom de la commande> : documentation des appels système
  - man 3 <nom de la commande> : documentation de la bibliothèque C

API Win32 de Windows et API d'Unix (1)

Unix	Win32	Description
fork	CreateProcess	Crée un nouveau processus
waitpid	WaitForSignalObject	Attend la fin d'un processus
execve	(rien)	CreateProcess = fork + execve
exit	ExitProcess	Termine l'exécution du processus
open	CreateFile	Cree un nouveau fichier (ou en ouvre un existant)
close	CloseHandle	Ferme un fichier
read	ReadFile	Lit des données depuis un fichier
write	WriteFile	Ecrit des données dans un fichier

API Win32 de Windows et API d'Unix (2)

Unix	Win32	Description
lseek	SetFilePointer	Déplace le pointeur de fichier
stat	GetFileAttributesEx	Trouve l'information sur un fichier
mkdir	CreateDirectory	Crée un nouveau répertoire
rmdir	RemoveDirectory	Supprime un répertoire vide
chdir	SetCurrentDirectory	Change le répertoire de travail courant
chmod	(rien)	
kill	(rien)	Win32 ne supporte pas les signaux
time	GetLocalTime	Trouve l'heure courante

Plan

1. **Introduction aux systèmes d'exploitation**
   - *Qu'est-ce qu'un système d'exploitation*
   - *Historique*
   - *Concepts de base*
   - *Interfaces d'un système d'exploitation*
2. Processus
3. Gestion des processus
4. Fichiers
5. Mémoire
6. Etude de cas

Références

- **Systèmes d'exploitation – 2ème édition**, A. Tanenbaum, Pearson Education, 2003.
- **Practical UNIX Programming**, K. A. Robbins, S. Robbins, Prentice Hall, 1996
- **Principe des systèmes d'exploitation des ordinateurs**, S. Krakowiak, Dunod, 1985.
- Ce cours a été conçu à partir d'autres supports :
  - Sacha Krakowiak, http://sardes.inrialpes.fr/people/krakowia/
  - Fabienne Boyer, http://sardes.inrialpes.fr/people/boyer/cours/SR/