Programmation Système

Cours 2

Concepts généraux des systèmes d'exploitation

Brice Goglin

Copyright

Plan

Objectifs et historique

Objectifs des systèmes d'exploitation

- Rendre le système plus facile à utiliser
  - Abstraction des périphériques
- Améliorer l'efficacité du système
  - Mettre les ressources à disposition efficacement
- Facilité d'évolution
- Protection des différents programmes
- Sécurité vis-à-vis des autres utilisateurs

Interface entre utilisateur et machine

- Interface uniforme d'accès aux périphériques
  - Détails techniques cachés
- Exécution de programmes
- Accès contrôlé aux fichiers
  - Montrer le contenu structuré des périphériques de stockage
  - Fournir des fonctions d'édition, développement, ...

Interface entre utilisateur et machine (2/2)

- Gérer les erreurs proprement
  - Signaler aux programmes les erreurs matérielles
  - Réagir en cas d'erreur logicielle
- Fournir des statistiques d'utilisation et fonctionnement
  - Permet à l'utilisateur de mieux adapter la configuration du système

Évolution

- *Serial Processing*
- *Simple Batch Systems*
- *Multiprogrammed Batch Systems*
- *Time Sharing*

Concepts généraux

Processus

- File d'exécution
  - Etat des registres
- Contexte mémoire propre
  - Exécution indépendante des autres processus
- Données privées au système d'exploitation
  - Pour manipuler le processus
- Exécution concurrente pour maximiser utilisation des ressources matérielles

Gestion de la mémoire

- Isolation des processus
  - Pas de collision entre leurs mémoires
    - Données et instructions propres
- Allocation et gestion transparente
  - Pas de contraintes sur le programmeur
- Programmation modulaire
- Partage de mémoire avec protection
- Stockage en mémoire persistante

Mémoire virtuelle

- Virtualisation des adresses mémoire manipulées par les processus
  - Illusion d'être le seul processus
- Découpage en pages (et/ou segments)
- Stockage par défaut de tout sur le disque
  - Rappel en mémoire des pages utilisées
- Partage de pages aisé
- Support matériel

Protection des données et sécurité

- Contrôle des autorisations
  - Accès aux données critiques du système
  - Accès et modification des données des utilisateurs
- Authentification des utilisateurs
- Survie à un problème technique

Ordonnancement et gestion des ressources

- Accès aux ressources équitable
  - En particulier pour les travaux de même type
- Différentiation de différentes classes de travaux
- Contrôle dynamique
- Efficacité
  - Utilisation du matériel et réactivité

Ordonnancement

- *Round-Robin*
- Priorités
- Priorités dynamiques
- Contraintes *deadline*
- Contraintes matérielles
  - *Elevator* et *Batch*
- Anticipation

Structure

Structure du système d'exploitation

- Noyau
  - Coeur
  - Pilotes de périphériques
  - Interface utilisateur
- Bibliothèques utilisateur de bas niveau
  - Appels système

Structure du système d'exploitation (2/2)

- Gestion de la mémoire au coeur du système
  - Processus
  - Systèmes de fichiers
  - Réseau
- Systèmes de fichiers et stockage
- Ordonnancement
- Communications entre processus
- Réseau

Noyaux modulaires

- Chargement/déchargement dynamique de code
  - Pilotes de périphériques
  - Fonctionnalités spécifiques
- Réduction du noyau initial
- Possibilité d'évolution dynamique
  - Sans redémarrage

Micronoyaux : Idées

- Noyaux monolithiques trop gros
  - Pas organisés, même avec des couches
  - Sécurité difficile à intégrer car trop d'interactions
  - Difficile à maintenir et faire évoluer
- Micronoyaux
  - Uniquement le strictement nécessaire
  - Tout le reste dans les processus serveurs dédiés

Micronoyaux (2/3) : Design

- Un serveur pour chaque tâche
  - Processus, gestion mémoire, ordonnancement, réseau, systèmes de fichiers
  - Passage de message entre applications et serveurs
    - Validés par le micronoyau
- Interfaces simples et uniformes
- Extensible, flexible, portable
- Fiable

Micronoyaux (3/3) : Performance

- Passage de message plus lent qu'appel direct de procédure du noyau ?
- Trop d'intéractions critiques entre différentes parties du système d'exploitation ?
  - Par exemple : Mémoire et Systèmes de fichiers
- Difficile de comparer avec noyaux monolithiques
  - Pas de vrai OS fonctionnel basé sur micronoyau

Avancées récentes

Multithreading

- Exécution simultanée de plusieurs *threads* dans le même processus
  - Travailler pendant qu'un *thread* bloque sur une I/O
- Ressources d'exécution propres
  - Pile et registres
- Espace mémoire partagé

SMP et NUMA

- Nouvelles contraintes sur l'ordonnancement
- Plusieurs processeurs
  - Accès concurrents
  - Affinité pour un processeur
    - Cache, TLB, ...
  - Affinité des threads d'un même processus
- Contraintes NUMA
  - Affinité pour certaines zones mémoire

Entrées/sorties asynchrones

- Matériel naturellement asynchrone
  - Soumission de commandes
  - Attente active de statut ou passive d'IRQ
- Remontée de l'asynchronisme jusqu'à l'application
  - Pas besoin de threads pour recouvrir les I/O

Systèmes distribués

- Grappes d'ordinateur
- Systèmes à image unique
- Coopération de différents noeuds
  - Avec ou sans maître
- Mémoire partagée distribuée
  - *Ping-pong* de pages
- Problème de synchronisation

Systèmes orientés objet

- Encapsulation des différents éléments
- Discipline pour les extensions modulaires
- Modifications sans menacer l'intégrité
- Facilite les systèmes distribués

Exemples

Windows

- Pseudo micronoyau
  - Beaucoup d'autres fonctions en mode noyau
- Très modulaire
  - *Executive* : Ensemble de *managers*
    - *I/O*, *Cache*, *Object*, *PnP*, *Power*, *Security*, *VM*, *Process*, ...
  - Couche d'abstraction des périphériques
  - Pilotes de périphériques
  - Fenêtrage et graphisme

Windows (2/2)

- *Local Procedure Calls*
  - Echange de message entre applications et managers
  - Modèle *Client-Server*
- Support SMP, threads et communications entre processus
- Design orienté objet
  - Polymorphisme

Unix

- Créé à la fin des années 60
- Design pour serveur et réseau
- Portable car rapidement écrit en C
- Tout est fichier
- Noyau (*kernel*) + Ensemble de bibliothèques et applications
  - Appels système (*System Call Interface*)

Unix (2/3)

- Noyau monolithique
  - Fonctionnalités communes
  - Gestionnaire mémoire
  - Périphériques bloc
  - Périphériques caractère
  - Ordonnanceur
  - Interface Vnode/VFS

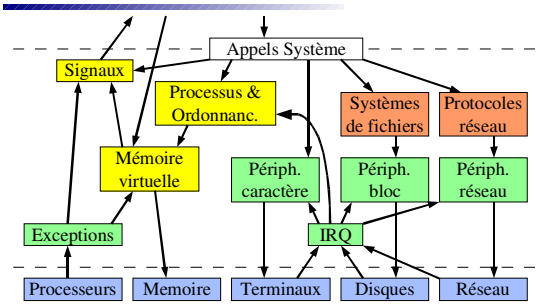
Unix (3/3)

- *System V Release 4*
  - Académique et commercial (AT&T et Sun)
- *Solaris*
  - Distribution commerciale (Sun) basé sur SVR4
- *Berkeley Software Distribution*
  - Très répandu dans le monde académique
  - Base de Mac OS X
- Beaucoup d'autres...

Linux

- Apparu en 1991
- Basé sur *Minix*
  - Pas trop cher pour usage personnel
- Libre
- Développement collaboratif sur Internet
- Supporte de nombreux architectures et périphériques matériels

Linux (2/4)



Linux (3/4)

- Chargement dynamique de modules noyau
  - Chargement automatique selon les besoins des applications
  - Hiérarchie basée sur dépendances de symboles
- Structure monolithique particulière
  - Seules certaines fonctions sont accessibles aux autres modules
  - Pas d'interface fixée

Linux (4/4)

- module_init, module_exit
- module_param
- insmod, rmmod, modprobe
- EXPORT_SYMBOL