Systèmes d'exploitation II
Réseaux et Système Avancés

Moufida Maimour <moufida.maimour@telecomnancy.eu>

TELECOM Nancy – 2ème année

2015-2016

Conception des systèmes d'exploitation

(SE ou OS, Operating System)

- Concepts et algorithmes généraux
- Etudes de cas (Linux 2.6, 4.4BSD, Système V, SOLARIS)

Objectifs

- Comprendre les principes de conception et la structure d'un SE
- Identifier quelques problèmes à résoudre lors de la conception des SE
- Comparer et évaluer les solutions proposées à ces problèmes

Pré-requis :

- Notions de base sur l'architecture des ordinateurs (PFSI),
- Pratique du langage C et du shell UNIX (CSH),
- Programmation système (RS)

Évaluation

- Un écrit

Contenu du module

1. Généralités : systèmes informatiques et systèmes d'exploitation
   - Rappel et compléments sur la structure générale des systèmes informatiques, quelques aspects de l'architecture du Pentium ;
   - composants d'un OS, conception d'un OS, organisation générale du noyau Unix.
2. Gestion de la mémoire
   concepts, impact de la liaison d'adresses, mémoire uniforme et hiérarchisée, le cas linux 2.6.
3. Gestion des processus et ordonnancement
   espace d'adressage, implémentation des processus, changement de contexte, algorithmes d'ordonnancement, les cas SOLARIS, HP-UX, 4.4BSD, Linux 2.6.

Bibliographie succincte

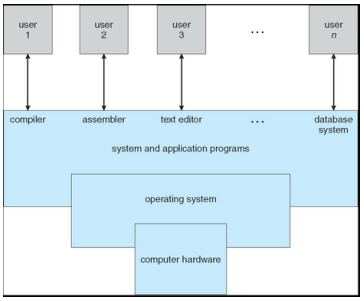
- A. Silberschatz, J.L Peterson and P.B. Glavin : Operating Systems Concepts (8th edition). `http://www.os-book.com/`, source de nombreuses figures de ce document.
- A. S. Tanenbaum, A. S.Woodhull : Operating Systems : Design and Implementation (3rd Edition)
- D. P. Bovet and M. Cesati : Understanding the Linux Kernel (3rd edition)
- M. J. Bach : The Design of the Unix Operating System (Prentice Hall)
- M. K. McKusik, K. Bostic, M. J. Karels et J. S. Quarterman : Conception et Implémentation du Système 4.4BSD

Première partie

Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
- Conception des systèmes d'exploitations
  Le noyau
  Structure simple : MS-DOS
  Structure en couches
  Systèmes monolithiques
  Micro-noyau
  Modules et noyaux modulaires
- Organisation générale du noyau Unix
  Architecture du noyau Unix
  Invocation des services système

Système informatique

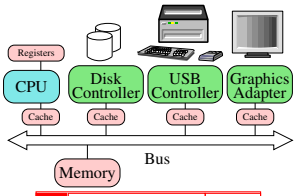


- Matériel : CPU, mémoire, périphériques
- OS : permet l'utilisation du matériel en optimisant ses ressources
- Applications : éditeurs de texte, compilateurs, base de données, jeux video . . .
- Utilisateurs : humains, machines, d'autres ordinateurs

Système informatique : le matériel

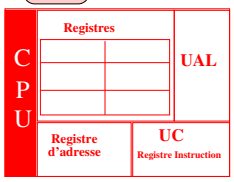
Système informatique moderne

- CPU (Central Processing Unit)
- Structures de stockage
- Périphériques d'E/S
- Communication via un **bus**



CPU (Central Processing Unit)

- ALU : opérations arithmétiques et logiques
- Registres : de données, d'adresse, SP, PC, . . .
- Unité de contrôle (UC) : A chaque coup d'horloge, elle ouvre et ferme des portes dans le but d'alimenter l'UAL en données et instructions et d'interdire la collision d'informations.



Système informatique : le matériel

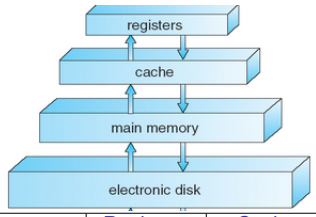
Sous-ensembles du bus

- Bus de données : véhicule les données
- Bus d'adresse : véhicule des adresses et conditionne la taille mémoire adressable
- Bus de commande : véhicule tous les signaux utilisés pour synchroniser les différentes activités : signaux d'horloge, R/W, interruptions . . .

Pentium : 64 lignes de données (E/S), 32 lignes d'adresses et quelques lignes de contrôle

Système informatique : le matériel

Hiérarchie de stockage



- taille,
- vitesse,
- coût,
- volatilité

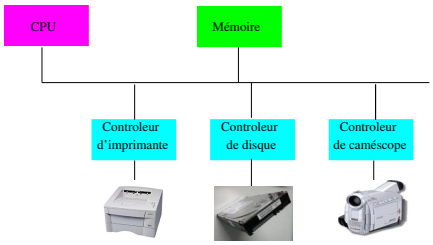
Level	Registers	Cache	Main Memory	Disk storage
Typical size	<1Kb	few Mb	few Gb	100s of Gb
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	5,000,000
Bandwidth (Mb/s)	20k - 100k	5k - 10k	1k - 5k	20 - 150
Volatile ?	Yes	Yes	Yes	No
Managed by	Compiler	Hardware	OS	OS
Backed by	Cache	Main Mem	Disk	CD or tape

Système informatique : le matériel

Périphériques d'E/S : device controllers

Device controller : la partie électronique du périphérique. Elle maintient un ensemble de registres :

- command registers (write-only) ;
- status registers (read-only) ;
- data registers (read/write).



CPU peut accéder à ces registres via des instructions d'E/S (in/out dans Intel Asm) ou via un mapping mémoire.

Pentium

Les registres

General-Purpose Registers

31 16	15 8	7 0	16-bit	32-bit
	AH	AL	AX	EAX
	BH	BL	BX	EBX
	CH	CL	CX	ECX
	DH	DL	DX	EDX
	BP			EBP
	SI			ESI
	DI			EDI
	SP			ESP

Remarque : les registres 8 bits 8088. Le registre AX 16 bits remonte au vieux 80286. Le registre 32 bits EAX n'existe qu'à partir du 80386.

- EAX : accumulateur,
- EBX : adressage indirecte,
- ECX : compteur,
- EDX : overflow,
- ESI et EDI (Source Index & Destination Index) : chaînes de caractères,
- EBP (Extended Base Pointer) : accès aux paramètres et variables locales,
- ESP (Extended Stack Pointer) : gestion de la pile,
- Compteur programme.

Pentium

Le registre d'état EFLAGS

- Il informe le processeur et le programmeur de l'état d'exécution du processeur, sur la nature du résultat d'une opération arithmétique ou logique.
- Instructions de branchement
- Certains bits comme le bit d'interruption permet au programmeur de déterminer le comportement du processeur face à certains événements.

Des registres spécialisés

- gestion de la mémoire virtuelle,
- le coprocesseur arithmétiques (FPU),
- le coprocesseur multimédia MMX,
- les extensions SIMD (SSE : Streaming SIMD Extension).

Pentium

Les registres de segments (1)

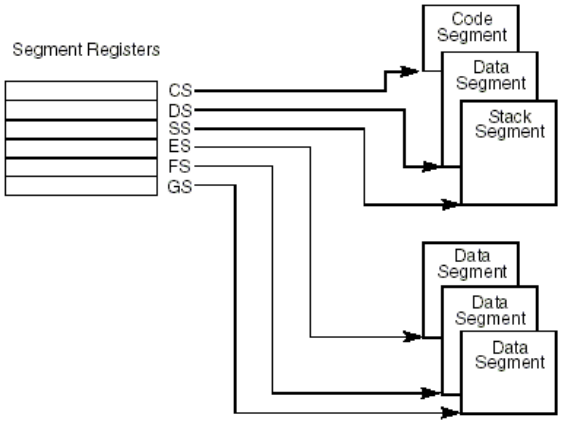
3 registres spécialisés

- cs (code segment), pointe sur le segment contenant les instructions du programme
- ss (stack segment), pointe sur le segment contenant la pile du programme en cours pour la gestion des sous-programmes et des interruptions.
- ds (data segment), pointe sur le segment contenant les données globales et statiques.

Il y a 3 autres registres plus généraux (ES,FS,GS).

Pentium

Les registres de segments (2)



Pentium

Principaux modes d'adressage en assembleur

- Adressage Registre

  addl %eax, %edx

  Ajouter le contenu de EAX à celui de EDX. Résultat dans EDX
- Adressage immédiat

  addl $10, %eax

  Ajouter 10 à la valeur contenue dans EAX. Résultat dans EAX
- Adressage indirect

  addl %edx, (%eax)

  EAX contient l'adresse de la valeur à ajouter au contenu de EDX. Résultat dans la case mémoire pointée par EAX.

Première partie

Généralités

C'est quoi un système d'exploitation

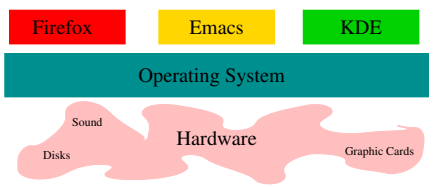
Un système d'exploitation est l'ensemble des programmes qui permettent l'exploitation du matériel :

Allocateur de ressources

- Gestion de toutes les ressources
- Gestion des conflits pour une utilisation efficace et équitable

Contrôleur

- Contrôle l'exécution des programmes pour éviter les erreurs et l'utilisation incorrecte de l'ordinateur



Evolution des SE

Systèmes monoprogrammés (monojob)

- Un seul programme, un seul utilisateur
- MS-DOS, systèmes embarqués
- Problème : usage inefficace des ressources matérielles. Pas de recouvrement des E/S.

Solution

- Permettre la coexistence de plusieurs programmes dans le système

Systèmes multiprogrammés (multijob)

- Plusieurs programmes peuvent être chargés en mémoire en même temps
  Si un processus se bloque (E/S), un autre pourra se voir attribué le processeur pour être exécuté
- Problèmes liés à la concurrence entre processus sont à résoudre

Systèmes multi-utilisateur (multiuser)

- apparrus avec l'arrivée des terminaux

Composants des systèmes d'exploitation

Principaux composants :

- Gestion des processus
- Gestion de la mémoire centrale
- Gestion du système d'entrée-sortie
- Gestion des fichiers
- Système de protection
- Gestion des réseaux
- Système interpréteur de commandes

Composants des systèmes d'exploitation

Système interpréteur de commandes

- interface entre l'utilisateur et le SE.
- inclu au noyau ou un programme spécial (Unix, MS-DOS)
- textuel (Unix) ou graphique (Windows)

Gestion des processus

- création (fork, exec) et terminaison (exit, kill) de processus système et utilisateur
- suspension et reprise des processus (wait, waitpid, sleep, pause)
- fournir les mécanismes de synchronisation et communication de processus (signaux, tubes, sémaphores, mémoire partagée, . . .)
- fournir des mécanismes pour le traitement des interblocages

Composants des systèmes d'exploitation

Gestion de la mémoire principale

Motivations

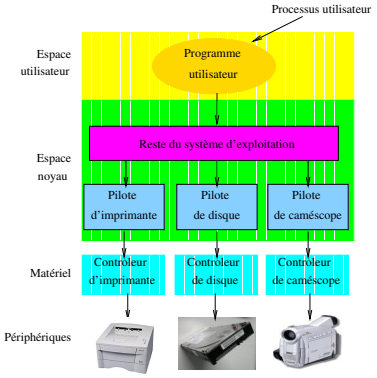
- La mémoire est en général, le seul dispositif de stockage auquel l'UC peut accéder directement
- Pour être exécuté, un programme doit être chargé en mémoire
- La multiprogrammation nécessite une gestion plus avancée de la mémoire afin de permettre la coexistence de plusieurs processus.

Le système d'exploitation doit :

- savoir en permanence, quelles parties de la mémoire sont en cours d'utilisation et par qui
- décider de quels processus doivent être chargés en mémoire lorsque l'on dispose d'espace mémoire
- affecter et désaffecter de l'espace mémoire

Composants des systèmes d'exploitation

Gestion du système d'E/S



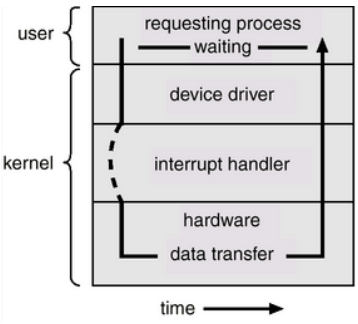
- Contrôler tous les périphériques d'E/S de l'ordinateur
- Fournir une interface entre les périphériques et le reste du système
  - indépendance du matériel : **sort** < **in** > **out**
  - désignation universelle (uniform naming) : **/dev/fd* /dev/mouse**

Mécanismes d'E/S

E/S synchrones (polling ou PIO)

- E/S bloquante
- Le processeur émet une commande à un périphérique et attend de manière active le changement du registre d'état
- Le périphérique traite la requête et change le registre d'état quand il a terminé et indique le succès ou une erreur

  ⇒ Utilisation CPU est faible

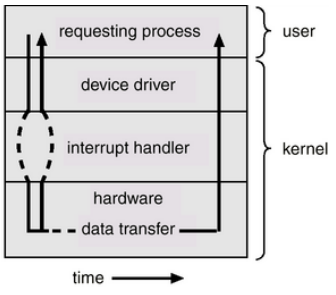


- Solution : interruptions

Mécanismes d'E/S

E/S asynchrones

- E/S non bloquante
- Le processeur émet une requête et continue son exécution
- Le périphérique traite la requête, une fois terminée, il émet une interruption
- Le processeur s'interrompt et interroge le périphérique pour savoir si la requête a été traitée avec succès



- Nécessaire dans le système pour multiplexage : ne pas bloquer le système pendant une E/S
- Peu habituel dans les applications : interface standard bloquante
- Recouvrement des E/S

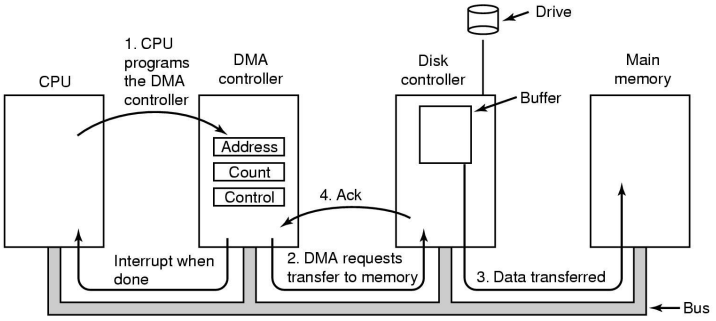
Mécanismes d'E/S

DMA (Direct Memory Access)

- Un moteur DMA gère le transfert de données entre la mémoire principale et un périphérique
- Le processeur envoie une requête de transfert à un moteur DMA :
  - type opération : lecture ou écriture
  - adresse du périphérique
  - adresse en mémoire centrale
  - longueur
- Le moteur DMA interrompt le processeur lorsque le transfert est terminé.
- Moteurs DMA placés sur les périphériques ou sur le bus d'entrées-sorties
- **Avantage :** amélioration de l'utilisation du CPU :
  - le transfert est juste initié par le CPU
  - pendant le transfert, le CPU peut faire autre chose

Mécanismes d'E/S

DMA (Direct Memory Access)



Mécanismes d'E/S

Quelle stratégie choisir

- Si le traitement par le périphérique est long ⇒ interruption : pas d'attente active
- Si la quantité de données à transférer est grande ⇒ DMA : pas de gaspillage du temps processeur
- Si la requête est simple à traiter ⇒ PIO

Composants des systèmes d'exploitation

Gestion des fichiers

l'une des composantes les plus visibles d'un système d'exploitation. Elle fournit :

- une vue logique uniforme du contenu des différents dispositifs de stockages (fichier, répertoire)
- la correspondance entre les fichiers (concept logique) et la mémoire auxiliaire (physique)
- la possibilité de sauvegarde des fichiers sur des supports d'information stables
- des primitives de :
  - création et suppression de fichiers (répertoires) : creat, mkdir, rmdir, . . .
  - manipulation de fichiers (répertoires) : open, read, write, lseek, close, stat, opendir, readdir, closedir,. . .

Composants des systèmes d'exploitation

Système de protection

Un système d'exploitation (multiprogrammé en particulier) dispose de ressources matérielles et logicielles qui doivent être protégées :

- Protection matérielle (mémoire, temps CPU, périphérique d'E/S, . . .)
- Protection logicielle (système de fichiers)

La protection matérielle : le matériel fournit des mécanismes matériels au niveau du processeur permettant :

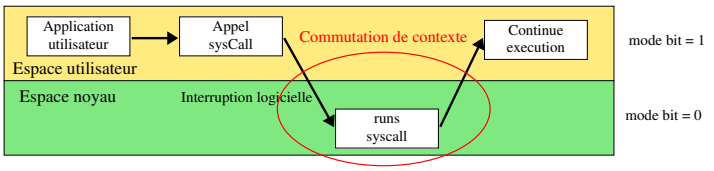
- protection intra-processus séparant le mode d'exécution utilisateur du mode d'exécution privilégié
  ⇒ Appels système pour l'exécution des instructions privilégiées
- protection inter-processus garantissant le fonctionnement multi-programmé
  ⇒ protection des espaces d'adressage des processus utilisateur et noyau

Système de protection

Protection matérielle : Modes d'exécution

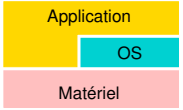
Un processeur actuel dispose d'au moins 2 modes d'exécution :

- mode privilégié (superviseur) associé au mode kernel (noyau) des systèmes Unix,
- mode non prévéligié associé au mode user (utilisateur) des systèmes Unix



Intel 80486 dispose de 4 niveaux d'exécutions :

- MS-Dos écrit pour 8088 ne possède pas de mode protégé. Tous les programmes utilisateur ont accès direct au matériel
- Les SE les plus récents comme Windows/Nt (Microsoft) et OS/2 (IBM) fournissent une meilleure protection



Système de protection

Exemples de protection matérielle (1/2)

Protection des E/S

- les instructions d'E/S sont considérées comme privilégiées.
- Un programme utilisateur doit passer par le système d'exploitation pour effectuer des E/S

Protection du vecteur et routines d'interruptions

- Éviter qu'un programme utilisateur ne puisse aller modifier le vecteur d'interruption :
  - Le programme pourrait mettre une adresse (la sienne ou celle d'un autre programme) dans une entrée de la table.
- Protéger les routines de traitement des interruptions d'être modifiées
  ⇒ Nécessité de la protection de la mémoire. Au moins la table et les routines de traitement des interruptions

Système de protection

Exemples de protection matérielle (2/2)

Protection de la mémoire

- Séparation de l'espace d'adressage des différents processus utilisateur ainsi que le noyau
- Mise en œuvre matérielle :
  - utilisation de 2 registres base et limite qui déterminent la fourchette d'adresse auxquelles un programme peut y accéder
  - seul le SE peut les charger en utilisant une instruction privilégiée spéciale.
  - une tentative d'accès en dehors de cet espace par le programme en mode utilisateur provoque un déroutement au SE.
- le SE possède un accès illimité à la mémoire du superviseur et à celles des utilisateurs.

Protection de l'UC

- s'assurer que le SE garde le contrôle. Exemple, éviter qu'un programme en boucle infinie ne rende jamais la main
- Solution : utilisation d'une horloge (temps partagé)

Première partie

Généralités

Conception des systèmes d'exploitations

Un système d'exploitation est un gros système et donc a besoin d'être conçu soigneusement pour fonctionner correctement.

Le noyau (kernel)

- le premier logiciel chargé en mémoire (hors gestionnaire d'amorçage)
- la partie fondamentale et la plus critique d'un système d'exploitation
- le gestionnaire des ressources de la machine qui permet aux éléments matériels et logiciels de communiquer entre eux, de fonctionner ensemble et de former un tout.

Fonctions de base d'un noyau :

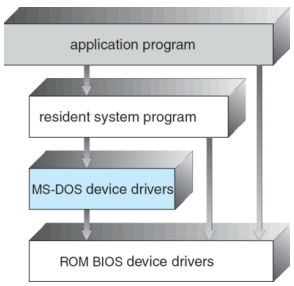
- chargement et exécution des processus
- gestion des entrées/sorties et du matériel en général
- interface avec les programmes de l'espace utilisateur
- gestion de la mémoire et ordonnancement

Conception des systèmes d'exploitation

Structure simple : MS-DOS

- Objectif : fournir un maximum de fonctions en un minimum d'espace
- non structuré en modules
- séparation non claire entre les différents niveaux

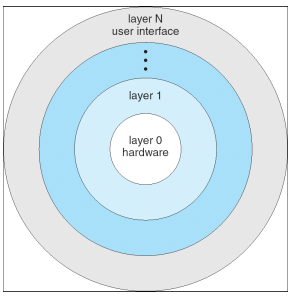
⇒ Structure en couches



Conception des systèmes d'exploitation

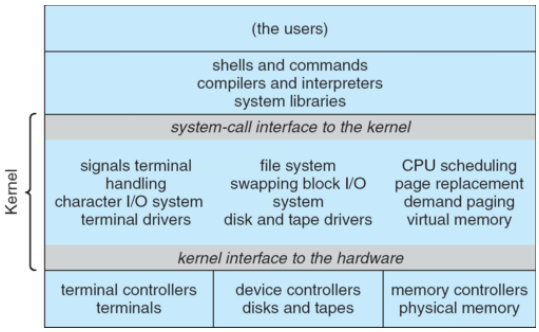
Structure en couches

- le SE est divisé en couches
  - la couche la plus basse : le matériel
  - la couche la plus haute : l'interface utilisateur
- les couches sont conçues de telle sorte qu'elles utilisent les services de la couche du niveau inférieur



Conception des systèmes d'exploitation

Structure en couches : UNIX



⇒ Unix (premières versions) : système monolithique

Conception des systèmes d'exploitation

Systèmes monolithiques

L'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation

Exemples : certains BSD et les anciennes versions de GNU/Linux et d'Unix

Avantages

- facilité de conception et de développement
- vitesse d'exécution

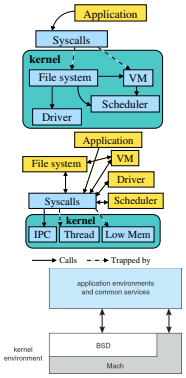
Inconvénients

- code volumineux, difficile à maintenir et à faire évoluer
- sécurité difficile à intégrer (interactions multiples)
- utilisation inutile de la mémoire (présence de tous les services même non utilisés)

Conception des systèmes d'exploitation

Systèmes à micro-noyau

- Le noyau ne contient que les fonctions essentielles :
  - quelques primitives basiques de la gestion des processus, un ordonnanceur simple, des E/S simples et un mécanisme de communication entre processus (IPC)
- Les autres services sont fournis par des programmes systèmes qui s'exécutent au-dessus du micro-noyau
- Exemples : Minix, Mac OS X, Hurd, Chorus (INRIA) . . .



Conception des systèmes d'exploitation

Systèmes à micro-noyau

Avantages

- modularité, portabilité, fiabilité et facilité de la maintenance
- permettent de concevoir des systèmes très généraux
- plus de sécurité en plaçant de nombreux services "à risque" en espace utilisateur
- le système est très configurable
- meilleure utilisation de la mémoire

Inconvénients

- performances réduites à causes des mécanismes lourds de communication (IPC)
- le grand nombre d'appels système (la plupart des services sont à l'extérieur du noyau)
- difficile de comparer avec les noyaux monolithiques car pas de vrai OS micro-noyau fonctionnel à ce jour

Conception des systèmes d'exploitation

Les noyaux modulaires

les parties principales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, les blocs de fonctions auxiliaires sont regroupés en différents modules qui peuvent être séparés (code et binaire).

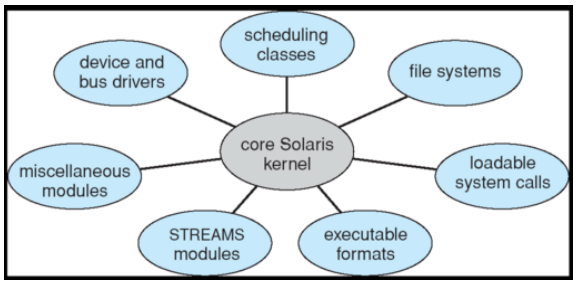
- bon nombre des avantages théoriques des SE micro-noyau sans pénaliser les performances
- un module implémente un système de fichiers, un pilote de périphérique, une fonctionnalité spécifique ou toute autre fonction de haut niveau d'un SE.
- Chargement, déchargement dynamique de code
- une fois inséré dans le noyau, un module est équivalent au code lié statiquement. Il est exécuté en mode noyau au nom du processus courant, comme toute fonction liée statiquement au noyau
- aucun passage de message n'est nécessaire lorsque les fonctions du module sont invoquées

C'est le cas de la majorité des systèmes actuels comme GNU/Linux, Solaris et la plupart des BSD

Conception des systèmes d'exploitation

Les noyaux modulaires

Solaris

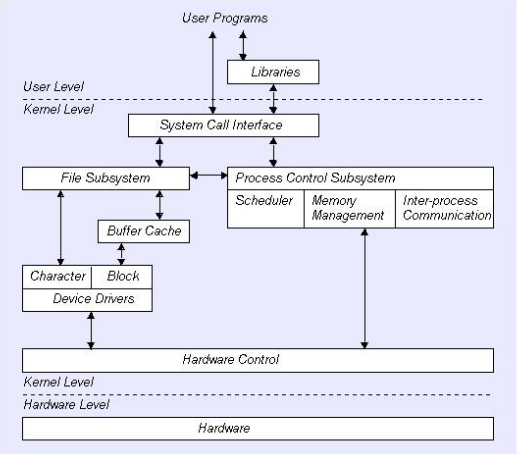


Première partie

Généralités

- Structure générale d'un système informatique
- Cas du Pentium
- Composants d'un système d'exploitation
- Conception des systèmes d'exploitations
  - Le noyau
  - Structure simple : MS-DOS
  - Structure en couches
  - Systèmes monolithiques
  - Micro-noyau
  - Modules et noyaux modulaires
- Organisation générale du noyau Unix
  - Architecture du noyau Unix
  - Invocation des services système

Architecture du noyau Unix



Architecture du noyau : approche descriptive

Le noyau constitué de 3 grandes parties :

- l'interface des appels système, interface entre les programmes utilisateur et le noyau
- le sous-système de gestion des processus
  - gestion des processus : création, terminaison, suspension, synchronisation et communication.
  - ordonnancement : traite la gestion du partage du temps et des priorités.
  - gestion de la mémoire : gère le partage des objets, la protection interprocessus, le swapping ou la pagination.
- le sous système de gestion des fichiers
  - gestion du buffer cache : gère l'allocation des tampons d'E/S.
  - gestion des fichiers : traite la protection, l'allocation de l'espace disque, la désignation des fichiers
  - gestion des périphériques : gère les fichiers en mode caractère et en mode bloc, l'accès aux périphériques, y compris aux réseaux.

Architecture du noyau : approche fonctionnelle

Le noyau UNIX est découpé en 2 grandes parties qui coopèrent pour le partage des ressources du système et pour la mise-en-oeuvre de certains services :

Partie supérieure

fournit des services aux processus utilisateurs en réponse aux appels système et aux exceptions.

- exécution synchrone en mode noyau pour pouvoir accéder à la fois aux structure de données du noyau et aux contextes des processus utilisateurs.

Partie inférieure

composée d'un ensemble de sous-programmes invoqués pour le traitement des interruptions matérielles

- des activités se déroulant d'une façon asynchrone et s'exécutent en mode noyau

Invocation des services système

Interruptions matérielles et exceptions

- Une interruption est provoquée par un signal provenant du monde extérieur au processeur, et modifiant le comportement de celui-ci. Le but est de le prévenir de l'occurrence d'un évènement extérieur :
  - fin d'une E/S, top d'horloge ...
  - 80x86 : 32-238. Linux utilise le vecteur 128 (0x80) pour les appels système.
- Une exception est un signal provoqué par un disfonctionnement du programme en cours d'exécution :
  - division par zéro, faute de page ...
  - 80x86 : 20 différentes exceptions 0..19. Les valeurs de 20 à 31 sont réservées par Intel pour le futur.
- Chaque interruption ou exception dispose d'un sous-programme (handler) qui prend en charge l'évènement correspondant : table de vecteurs d'interruptions ou IDT : Interrupt Descriptor Table dans le langage Linux.

Invocation des services système

Traitement d'une exception ou d'une interruption

- Arrivée de l'interruption/exception
- Sauvegarde du contexte actuel (PC ...) en utilisant la pile noyau
- Accès à la table des vecteurs d'interruptions pour déterminer l'adresse du sous-programme de l'interruption (le handler) et chargement du PC avec son adresse
- Exécution du sous-programme en mode noyau
- Rétablissement de l'ancien contexte et reprise de l'ancien programme en mode utilisateur

Invocation des services système

Traitement des exceptions sous Linux

- La plupart des exceptions issues du CPU sont interprétés par Linux comme des cas d'erreurs : le noyau envoie un signal au processus qui a causé l'exception.
  - Exemple. division par zéro : envoi signal SIGFPE.
- Certaines exceptions entrainent la réexécution de l'instruction en cause.
  - Exemple. défaut de page.
- Handler d'une exception :
  1. Sauvegarde du contenu de la plupart des registre dans la pile noyau (assembleur)
  2. Traitement de l'exception (fonction C)
  3. Quitter le handler en invoquant la fonction ret_from_exception

Invocation des services système

Traitement des interruptions sous Linux

- Pas de signal envoyé au processus en cours.
- Interruptions d'horloge
- Interruptions d'E/S
  1. Sauvegarde la valeur de l'IRQ et le contenu des registres dans la pile noyau
  2. Envoi d'un ACK au PIC, lui permettant de traiter d'autres interruptions
  3. Exécution des routines de traitement d'interruption (ISR : Interrupt Service Routines") associées aux périphériques qui partage la ligne IRQ.
  4. Invocation de la fonction ret_from_intr()

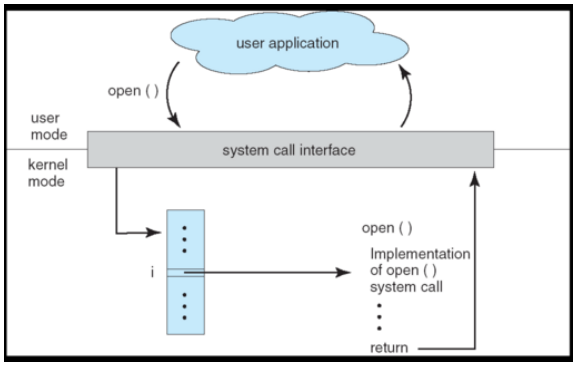
Invocation des services système

Les appels système

- L'interface entre le SE et les programmes utilisateurs est définie par l'ensemble des appels système fournis par ce dernier.
- Un appel système peut être vu comme un appel d'une fonction classique effectuée en mode noyau.
- Un appel système est généralement réalisé à l'aide d'une interruption logicielle avec un déroutement vers un emplacement spécifique dans la table des vecteurs d'interruptions.
- Une interruption logicielle est déclenchée par un programme à l'aide d'une instruction spéciale (trap, syscall)
- Il n' y a pas de changement de processus (préemption)
- Le handler est exécuté en utilisant les ressources du contexte du processus interrompu (la pile noyau).
- Des informations nécessaires à la requête peuvent être passées (par registres, pile ou mémoire)

Invocation des services système

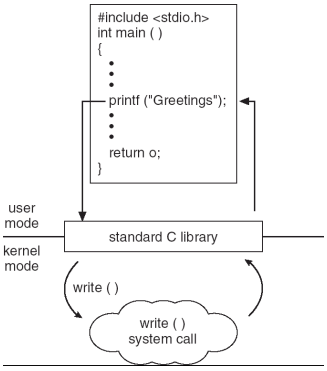
Implémentation des appels système



Invocation des services système

Bibliothèque standard C

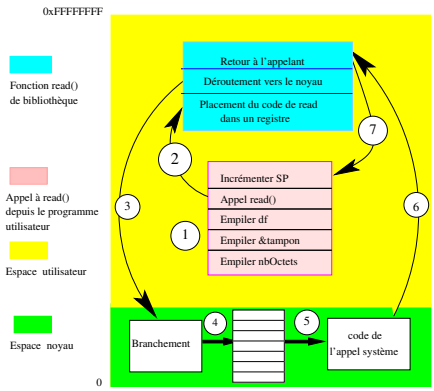
- le code d'un appel système est souvent en assembleur, mais une fonction de bibliothèque de fonctions en C est souvent fournie.



Invocation des services système

Un exemple d'appels système : read()

count = read(df, tampon, nbOctets)



Invocation des services système

Appels système sous Linux