Système d'exploitation

Entrées sorties
Mémoire

Florent Devin

Ecole Internationale des Sciences du Traitement de l'Information

Entrées sorties physiques

Plan

Rappel

Caractéristiques

- Vu en ADO
- Deux façons pour le transfert de données
  - Vers un périphérique (contrôleur : solution INTEL)
  - Vers une adresse particulière (Motorola)

Contrôleurs

Utilisation

- Chaque périphérique connecté à un contrôleur
- Permet la communication entre l'UC et le périphérique
- Communication via un bus
- Possibilité de connecter plusieurs périphériques à un contrôleur

DMA

Direct Memory Acces

- Utilisation d'un DMA pour accélérer le traitement
- Zone de mémoire réservé pour faire l'échange
- Contrôleur communique via cette zone mémoire
- Utilisation du bus, lorsqu'il n'est pas utilisé
- Microprocesseur : gère l'ensemble des transfert
- Émission d'un signal par le contrôleur lors de la fin du transfert

Plan

Introduction

Introduction

- Pour éviter d'avoir à gérer
  - la multitude de périphérique
  - l'évolution de ceux-ci
- Création de périphériques virtuels
- Sous unix : fichiers
- Association d'un *device driver*

Opérations possibles

Opérations

- Ouverture : `open`
- Fermeture : `close`
- Lecture : `read`
- Écriture : `write`
- Contrôle : `seek`

Mémoire

Plan

Rôle de la mémoire

Fonctionnalités attendues

- Abstraction : translation d'adresse
- Arbitrage inter processus : protection, allocation, partage
- Extension : gestion du matèriel

Plan

Translation

Translation

- Processus : ne connaît pas à l'avance sa localisation
- Localisation variable en cours d'exécution
- Distinction entre adresse logique et adresse physique
- $@_{physique} = f(@_{logique})$
- Fonction de translation : physique ou logicielle

Solution logicielle

Solution logicielle

- Utilisation dans le processus que des adresses relatives
  - Besoin de calculer l'adresse physique à chaque accès
- Utilisation d'une liste d'adresse absolue modifiable par le système au chargement du processus, ou pendant le changement de localisation

Solution physique

Solution physique

- Selon le mode du processus : mode utilisateur
  - Adresses considérées comme logique, utilisation d'un MMU (Memory Management Unit)
- Mode superviseur : adresses sont des adresses physiques

Plan

Problématique

Arbitrage

- Empêcher les processus d'accéder aux ressources des autres processus
- Empêcher la fragmentation de la mémoire lors de la (ré)allocation
- Permettre l'utilisation de mémoire partagée

Protection

Protection

- Utilisation de deux registres : base, limite
- Si le processus manipule des adresses externes alors SIGSEGV

Autres fonctions

Autres fonctionnalités

- Pas de réponse simple
- Besoin d'introduire la pagination et/ou segmentation

Plan

Introduction

Introduction

- *frame* : division de la mémoire en blocs de taille fixe
- *page* : division de la zone mémoire d'un processus en taille identique à la taille de la *frame*
- Processeur : registre dédié à la gestion de la table de correspondance

Utilisation

Utilisation

- Chargement uniquement des pages qui sont utiles
- Adresses processus : contiguës
- Adresses physiques : peuvent être disjointe
- Table de correspondance : résout le problème de translation
- Permet de simplifier la protection : une frame un processus
- Fragmentation effective, mais pas gênante
- Permet une gestion plus fine lors du passage en swap

Gestion de la pagination

Gestion

- Besoin d'un gestionnaire de page
- Problème : définir un gestionnaire efficace, et peu coûteux
- Implantation d'algorithme de changement de page
- Deux principes :
  - Localité : probabilité grande de réutilisation d'une page récente
  - Accessibilité : stocker dans le cache les données utiles
- Plusieurs algorithmes : FIFO, LIFO, NRU, LRU, ...

Processus et mémoire

Rappel

Processus et mémoire

- Plusieurs processus en mémoire
- Quand trop de processus, utilisation du *swap*
- Taille d'un processus : peut être plus grande que la mémoire
- Seul solution possible : utilisation de page, ou segment
- Permet de ne conserver que les "morceaux" utilisés
- Besoin d'un gestionnaire de mémoire pour permettre de tout gérer

Plan

Multiprogrammation

Rappel

- Plus il y a de processus en mémoire
- Plus le processeur est utilisé
- Nécessité d'augmenter la mémoire : pas toujours faisable (coûts)
- Division de l'espace mémoire en n partitions (de taille inégale éventuellement)

Gestion de la mémoire avec *n* partitions

Deux méthodes

- Une file d'attente par partition
  - Perte de place (gestion de la file)
  - Possibilité d'avoir des zones inexploitées
- Une file d'attente globale
  - Affectation selon la position dans la file, ou selon l'espace mémoire libre

Problème de translation

Problème de localisation

- Processus : endroit quelconque de la mémoire
- Pas de concordance entre l'adresse du programme, et physique
- Besoin de connaître l'adresse de base du processus (permet de réaliser un décalage)
- Besoin de connaître la taille de la mémoire adressable pour éviter l'écriture dans un autre processus : utilisation de clé de partition ou de registre limite
  - Clé : permet de connaître l'ensemble des partitions "écrivables"
  - Registre : permet de connaître l'@ fin théorique de la mémoire

Plan

Introduction

Introduction

- Nombre de processus $>$ Nombre de partitions
- Besoin de simuler la présence des processus en mémoire
- Utilisation de *swap* (appelé aussi recouvrement)
- Utilisation de partitions de taille variable : évolution des processus
- Nécessite un algorithme plus complexe (allocation et libération)

Gestion du swap

Rôle du swap

- Gérer par le processus 0
- Allocation du swap
- Transfert vers la mémoire
- Transfert hors de la mémoire

Allocation du swap

Allocation

- Gestion différente d'un disque
- Ensemble contigu de blocs de taille fixe
- Utilisation des fonctions
  - `swapalloc (int taille);` : demande d'unités d'échange
  - `swapfree (int adresse, int taille);` : demande de libération d'unités
- Possibilité d'avoir plusieurs unités d'echange
- Création et destruction dynamique

Transfert vers la mémoire

Transfert

- Examen des différents processus prêt présents sur le disque
- Choix du plus ancien
- Transfert de celui-ci : allocation de mémoire, lecture depuis le disque, libération de l'espace utilisé en swap
- Exécution jusqu'à
  - Plus de processus prêt sur le disque
  - Plus de place en mémoire

Transfert hors de la mémoire

Transfert

- Transfert en priorité des processus bloqué, puis les prêts
- Aucun transfert de processus Zombie, ni verrouille par le SE
- Transfert d'un processus prêt : uniquement si il est présent en mémoire depuis un certain temps (en général 2s)
- Si pas de possibilité : "re-scan" toutes les secondes

Problème

Étreinte fatale

- Tous les processus en mémoire sont “endormi”
- Tous les processus “prêt” sont transférés en swap
- Plus de place en mémoire
- Plus de place en échange sur le disque

Deux autres possibilités

Autres types de transfert

- Création d'un fils : possibilité de créer e fils en swap, lorsqu'il n'y a plus d'espace
- Accroissement de la taille d'un processus : Transfert du processus en swap avec l'allocation adéquate; au retour dans la mémoire le processus occupe plus de place