Sem 2

1 cours

PSE

Repertoire :
- Liste les type de fichiers (entrées)
- Deux entrées particulières : "." courant et ".." parent

Fichier = ensemble de blocs :

I Node | Taille | ...

mkdir (chemin, mode)
rmdir (chemin
link (fichier

Lien sur un fichier

- Lien mat. : même I-Node
  Lien sym : même I-Node ≠

Droits spéciaux

Substitution du propriétaire : suids
s'applicable aux fichiers exe garan
les droits du propriétaire pendant le
temps de l'exécution. (l'utilisateur
doit avoir le droit d'exécution). s remplac
le x du propriétaire
Substitution du grp du prop : pareil
mais avec les droits du groupe.

Stickey bit : permet d'empêcher la suppression des fichiers par les autres utilisateurs (sachant qu'ils ont les droits dans le répertoire).
remplace le x de other

ACL : liste de contrôle d'accès :

- En système : système très fin de gestion des droits
- Réseau : black list d'accès à un pare feu.

→ On associe à chaque objet (fichiers) tous les domaines (Réseau associé à un département de l'entreprise) qui ont le droit d'y accéder.

→ Utilisation en ligne de commande :
- setfacl : set file acces control list
  - -m modifier

- getfacl : donne des infos sur les droits donnés par la facl.

Processes & Thread

Processes = programme en cours d'exec
actif passif

↳ segments de mémoire, section donnée

Quand on change un process :
- Activité actuelle : compteur

- pile (stack) : variables
- heap (tas)

Les modèles de processus

1) Compteur ordinal bascule entre les processus les uns après les autres (en mémoire)

2) 4 compteurs ordinaux logiques (partage logique du CPU)

3) 1 compteur ordinal bascule entre les processus en fonction d'une fraction de temps alloué à chaque processus. (Répartition du temps CPU).

Events qui créent des processus :

- Init du système
- Appel système (fork())
- Requête utilisateur (saisie commande).
- Traitement pas lots (mainframe)

Fin d'un processus.
- Arrêt normal : appel système exit();
- Arrêt pour erreur : nullpointer, div by 0
- compilation d'un prog inexistant
- Arrêt par un autre processus : appel

Mystère fichier.

Etat des processus :
- new : en cours de création.
- running : exec des instructions
- waiting : attend un event (saisie clavier)
- ready : prêt à être affecté à un CPU
- terminated : fini son exécution
- Zombie : terminé mais ressources non libérées

Thread

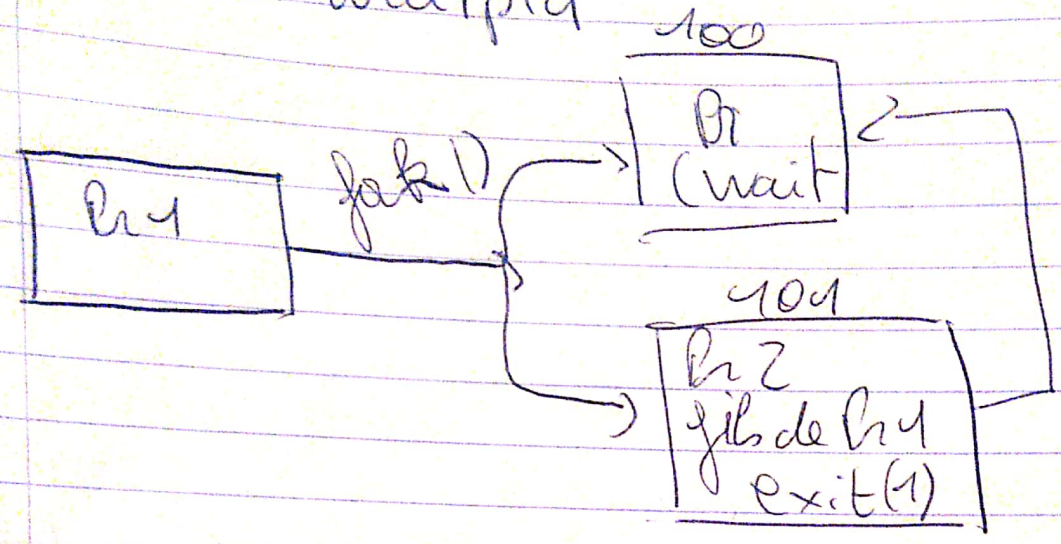
Unité de base de l'utilisation d'un processus. Permet de diviser une tâche entre divers processus. "sous processus".

Processus multi-thread :
même code, données, fichiers
registres et Stack ≠

Avantage sur les processus :
→ 100 x plus rapide à créer
→ plusieurs activités sur un processus
→ communication simple
→ partage de ressource
→ exploitation multicore

Sem 4 — PSE — 1 Amphi

wait et waitpid




```
int status_fils;
pid_t pidfils;
pidfils = wait (&StatusFils)
```

↑ 101

Fonctions Execl()

execl : changer l'image du proc en cours d'exec pour un nouveaux programme

Sem 5

PSE 1 cours

Communication inter processus

Tubes :
- Assimilé à un fichier virtuel ouvert en lecture/écriture
- Pas représenté sur le disque

Sémaphores :
- Synchro et protec° des sections critiques des processus.
- Contrôle l'accès à une ressource partagée par plusieurs processus en associant cette ressource avec un feu

Sémaphore à compteur
un struct qui contient :
- une variable (la valeur)
- une file d'attente
- P(sem) Prokeren → décrémenter la valeur du sémaphore et bloque éventuellement le proc appelant
- V(sem) Verhogen : incrément + réveil d'un proc (action atomique)
- initialisation.

Sémaphore binaire (mutex)
sa valeur ne peut être que 0 ou 1
il est init à 1, vérouille une ressource critique pour un processus

Faire attendre dormir :
sleep(), wakeup()

Parfois le consommateur test et trouve 0 mais il est exécute ensuite le produ
il y a donc perte du signal wakeup.

Solution : utiliser des sémaphore pour mémoriser les wakeups
-> Sémaphore à compteurs pour la synchro (full et empty)
-> Sémaphore binaire pour mutex sur le compteur. (exmu)

full.
- Init à 0 avec list vide
-
-

empty :
-
-
-

exmu
-

Sem 5 — PSE — 2 cours

Signaux

3 possibilité à la réception :
- ignorer
- capturer et utiliser un gestionnaire qu'on a spécifié
- accepter l'action prédéfinie

fonctions de gestion des signaux :
- signal, sigaction (def gestionnaire)
- sigprocmask (bloquer des signaux)
- sigpending (lister signaux bloqué)
- sigsuspend (suspendre le proc)

Émettre des signaux
- raise (envoyer signal processus courant)
- kill
- killpg

PSE | Cours

→ Le SE doit gérer l'utilisation des différentes mémoires

Adresse logique / physique

Adresse symbolique que [CPU] →

Logique : Générée par le CPU (PDV du processus)
Physique : Adresse vue par la mémoire

Pour avoir plus espace prog en mémoire
↳ espace d'adressage
- propre à chaque processus
- Déterminer l'ensemble des adresses légales.
- La base : la plus petite adresse légal
- limite

Registre de base et de limite

Le registre de base contient la première adresse légal et le registre de limite contient la dernière adresse légale.

⚠

→ Convient si la mémoire est assez grande pour contenir tous les processus

Allocation de la mémoire :

- contigüe : chargement à la suite de chaque processus. → besoin de protection (registre de base / limite)

- First fit : trouver 1er bloc libre qui peut contenir le prog

Best fit : - On parcours toute la mémoire pour trouver le plus petite bloc qui peut contenir le prog

- Worst fit : pareil mais plus grand

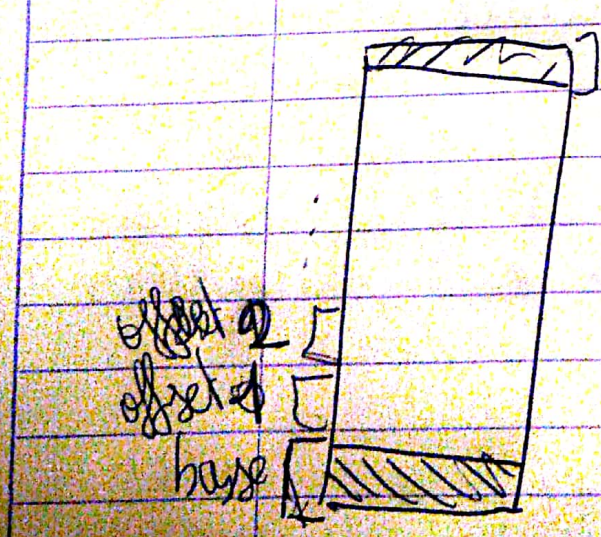
⚠ PB : fragmentation de la mémoire

↳ First fit est inefficace

SOLUTION : Segmentation

Mappage entre la mémoire logique et physique. On crée un segment pour chaque contexte d'exécution.

adresse logique : <numSeg, offset>



on vérif que d est legal (sécurité)

Inconvénient de segmentation :
- Fragmentation externe : blocs octets libres non contigüs
- Temps d'accès long et donc defrag impossible.

Pagination

- Découper la mémoire physique en bloc de taille fixe : frames (cadres de page) et découper la mémoire logique en des blocs de même taille : page
- Quand un processus est exec, les pages sont chargées dans les frames dispo

Taille adressage logique : $2^{\&m}$ et taille de page 2^n :

→ $m-n$ bits plus significatifs pour désigner la page

→ n bits pour l'offset

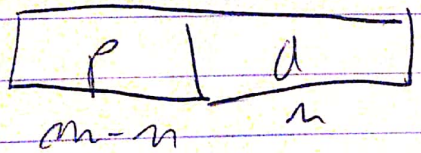


Table de page

Sem 7

PSE

1 Cours

Mémoire virtuelle:

Exécuter des processus qui ne sont pas entièrement en mémoire.

↳ les prog peuvent être plus grand que la mémoire réelle physique.

Swapping: Un processus peut être transféré sur le disque et rechargé dans la mémoire (le processus a besoin de ressources qui ne sont pas dispo et attend).
Mais le swapping prend un tps important

Pagination à la demande: On déplace des pages du processus qui ne sont pas utilisées pour le moment.

Page valide/invalide: sur la mémoire/sur le disque.

→ Quand on veut charger une page et que la mémoire est pleine, il y a plusieurs politique:
- FIFO
- OPTIMAL
- Least Recently used
- Least Frequently used

Scanned by CamScanner