Examen Intra

IFT-2245

February 10, 2018

Directives

- Documentation autorisée: une page manuscrite recto.
- Pas de calculatrice, téléphone, ou autre appareil électronique autorisé.
- Répondre sur le questionnaire dans l'espace libre qui suit chaque question. Utiliser le verso de la page si nécessaire.
- Chaque question vaut 5 points pour un total maximum de 25 points.
- Les questions ne sont pas placées par ordre de difficulté.

0 Nom et prénom (1 point de bonus)

Écrire son nom et prénom et son code permanent en haut de chaque page.

1 Synchronisation

Soit un système où vous avez accès à des *mutex* (ou *sémaphores binaires*) qui offrent deux opérations:

```
void acquire (mutex *m);
void release (mutex *m);
```

Les utiliser pour compléter le code ci-dessous des sémaphores entiers:

```
typedef struct semaphore {
  unsigned int value;


} semaphore;

void P (semaphore *s)
{









}

void V (semaphore *s)
{









}
```

2 Ordonnancement

Soit un système d'exploitation de style Unix utilisé pour un ordinateur de bureau. L'ordonnanceur utilise un système de priorités dynamiques, où chaque niveau de priorité a sa propre queue. Chaque niveau de priorité utilise un quantum de temps qui est le double de celui de la priorité antérieure et le noyau change la priorité des processus en observant leur comportement.

1. Donner les 2 plus importants critères de performance que l'ordonnanceur aimerait optimiser.

2. Quand l'ordonnanceur diminue-t-il la priorité d'un processus?
   Quand l'ordonnanceur augmente-t-il la priorité d'un processus?

3. Si tout se passe bien, quel genre de processus devrait obtenir une priorité élevée, et quel genre obtiendra une priorité basse?

4. Soit un processus de simulation de qualité de l'air qui prend beaucoup de temps et de mémoire, et que l'utilisateur laisse tourner pendant qu'il développe frénétiquement la version suivante dans son éditeur préféré (Emacs, bien sûr). Ce processus de simulation utilise (très activement) deux fois plus de mémoire virtuelle que la mémoire physique disponible. Quel niveau de priorité le noyau va-t-il donner à ce processus et pourquoi?

5. On ajoute un concept de *priorité statique souple* qui ne fixe pas un processus dans un niveau de priorité, mais qui doit influencer l'algorithme de priorité dynamique pour augmenter/diminuer la priorité dynamique que le noyau affectera au processus.
   Comment le noyau peut-il utiliser une telle priorité statique pour qu'elle aie l'effet désiré?

3 Traduction d'adresses

Soit un processeur où les adresses occupent 32bits, la taille des pages est de 1KB et qui utilise une table de pages hiérarchique classique, c'est-à-dire où chaque (sous-)table ne peut pas être plus grande qu'une page.

1. Combien de niveaux y a-t-il dans la hiérarchie?
2. Montrer pourquoi, en expliquant quels bits d'une adresse logique sont utilisés pour l'*offset* dans une page, et lesquels servent à indexer chaque niveau de la table de pages.
3. Donner la taille des (sous-)tables de chaque niveau.
4. Mentionner 1 avantage et 1 inconvénient de l'utilisation de tables hiérarchiques plutôt qu'une table à un seul niveau.
5. Décrire de manière concise et précise (i.e. du pseudo-code C) comment obtenir l'adresse physique qui correspond à une adresse logique.

4 Mémoire virtuelle

Soit un système d'exploitation classique qui fait de la pagination sur disque (i.e. qui stocke les pages inutilisées de mémoire virtuelle sur le disque).

1. Pourquoi ne *peut*-il pas utiliser l'algorithme de remplacement LRU (qui évince la page la moins récemment utilisée)?
2. Expliquer en détail comment il peut approximer LRU.
3. Pourquoi ne *veut*-il pas utiliser l'algorithme de remplacement LRU?
4. Quel type de *localité* est nécessaire pour que LRU fonctionne bien?
5. Quel est l'autre type de localité? Indiquer deux manières qu'un système de mémoire virtuelle peut tirer profiter de cette autre forme de localité.

5 QCM

Après `fork`, les deux processus partagent leurs mémoires	oui ☐	non ☐
Après `fork`, les deux processus partagent leurs piles	oui ☐	non ☐
Après `fork`, les deux processus partagent leurs descripteurs de fichiers	oui ☐	non ☐
Lorsque le parent de P meurt avant P, le processus P est un zombie	oui ☐	non ☐
Pour avoir une étreinte fatale, le graphe d'allocation de ressources doit contenir un cycle	oui ☐	non ☐
Les *spinlocks* n'ont de sens que dans un système monoprocesseur	oui ☐	non ☐
Avec seulement 2 niveaux de priorités, le problème d'inversion de priorité est impossible	oui ☐	non ☐
Un système multiprocesseur ne peut pas inhiber les interruptions	oui ☐	non ☐

Quels algorithmes d'ordonnancement peuvent résulter en une situation de famine	RR ☐	FCFS ☐	SJF ☐	Prio ☐
Quel algorithme donne une attente moyenne optimale	RR ☐	FCFS ☐	SJF ☐	Prio ☐