UNIVERSITÉ BORDEAUX I — U.F.R. Mathématiques et Informatique
Session de Février 1995 — J.Bétréma

Maîtrise d'Informatique
Systèmes d'exploitation 1
(Durée: 2 heures)

Documents interdits.

Le sujet comporte quatre exercices indépendants ; il est long, et le barème en tiendra compte : travaillez calmement, sans essayer de forcément tout traiter. Lisez entièrement l'énoncé d'un exercice, avant de commencer sa résolution.

1 Fichiers

La gestion des fichiers dans le système MS-DOS présente les deux particularités suivantes :

1. Une table (la FAT) indique, pour chaque bloc, le bloc suivant, de telle sorte que les blocs d'un fichier forment une liste chaînée.
2. Une entrée de répertoire comporte, outre le nom du fichier, ses attributs (date, droits d'accès, etc...), et le numéro du premier bloc.

Expliquer brièvement les inconvénients de chacun de ces choix, et comment UNIX y remédie.

2 Sémaphores

Expliquer en quelques lignes les principes de la synchronisation par sémaphores d'un processus producteur et d'un processus consommateur, avec un tampon de taille n.

3 Ordonnancement de processus

On considère trois processus simultanés P_1, P_2, P_3 . P_1 exécute un traitement de fichier, pendant lequel alternent des calculs, chacun d'une durée $\gamma_1 = 10$ ms, et des lectures de blocs, qui durent 20 ms, pendant lesquelles P_1 est bloqué. Le fonctionnement de P_2 est analogue, sauf que la durée de chaque phase de calcul, entre deux lectures de blocs, vaut $\gamma_2 = 30$ ms. P_3 n'effectue que des calculs, sans entrée-sortie.

L'ordonnanceur utilise l'algorithme du tourniquet, avec un quantum de temps $q = 200$ ms. On suppose enfin que le changement de contexte, lors du remplacement du processus élu par un autre, dure 2 ms ; on néglige la durée des autres interventions du système d'exploitation.

Au départ (instant 0), les processus P_1, P_2, P_3 se trouvent dans cet ordre dans la file des processus prêts, et P_1, qui vient d'être élu, commence son exécution.

1. (a) Décrire en détail le début de l'exécution entrelacée de ces trois processus : spécifier chaque intervention du système d'exploitation, avec la raison de celle-ci, et chaque changement d'état d'un processus.

   Note : cette exécution simultanée est censée être infinie; stopper la description dès qu'elle est suffisamment significative.

   (b) Calculer, pour chaque processus P_i, l'instant exact $t_i(k)$ où il est élu pour la $k^{\text{ème}}$ fois.

   (c) Calculer le débit utile du disque, c'est à dire le nombre de blocs utiles lus par seconde (arrondir la réponse à l'entier le plus proche).

   Note : on suppose que P_1 et P_2 ne lisent qu'un bloc à la fois.

   (d) Calculer le pourcentage de temps pendant lequel l'UC est attribuée à P_3 .

2. On suppose dans cette question que l'ordonnanceur utilise des files de priorité : la priorité de P_1 est égale à celle de P_2, et supérieure à celle de P_3.

   Chaque file est traitée suivant l'algorithme du tourniquet, et lorsqu'un processus est élu, l'UC lui reste attribuée jusqu'à épuisement de son quantum de temps, sauf si le processus se bloque.

   Répondre aux mêmes questions que précédemment (pour la partie (a), indiquer seulement les changements d'états des processus). Commenter les résultats.

   Donner les conditions sur γ_1 et γ_2 (durées des calculs entre deux lectures de blocs), qui entraînent un phénomène de famine pour P_3.

3. Reprendre la question précédente, en supposant cette fois que l'UC est retirée à un processus élu dès qu'il existe un processus prêt plus prioritaire. Commenter les résultats.

4. Dans le système UNIX, les priorités sont dynamiques : expliquer ce que cela signifie, et le principe de calcul de ces priorités. Dans un cas similaire à celui étudié ici, quel est le résultat de cette stratégie?

4 Gestion de la mémoire

Un processus traite une matrice m, de taille 128×128, dont chaque entrée est un réel en double précision (8 octets). Le langage de programmation utilisé est C.

1. En appelant b l'adresse de la base de m, calculer l'adresse de $m[i][j]$.

2. On suppose que la mémoire est paginée, et que chaque page est de taille 4Ko. On exécute la double boucle suivante :

```
for (i = 0; i < 128; i++)
  for (j = 0; j < 128; j++)
    { ... x = m[i][j]; ... }
```

et on suppose qu'aucune page n'est, initialement chargée en mémoire. Calculer le nombre de défauts de pages provoqués par cette exécution.

On suppose que les instructions `{ ... x = $m[i][j]; ... }` sont exécutées en $10\mu s$, et qu'une page est chargée en 10 ms. On désigne par t_1 le temps de calcul proprement dit, et par t_2 le temps de chargement des pages. Calculer le coefficient de dégradation t_2/t_1, dû à la pagination.

Si le nombre de pages physiques est insuffisant, commenter, dans ce cas, l'influence de l'algorithme de remplacement de pages.

3. On exécute la double boucle suivante :

```
for (j = 0; j < 128; j++)
  for (i = 0; i < 128; i++)
    { ... x = m[i][j]; ... }
```

   Donner la suite des références de pages (de données) qui correspond à cette exécution ; on supposera pour simplifier que l'adresse de base b est nulle.

   On suppose la mémoire physique disponible limitée à 16 pages. Décrire les remplacements de pages effectués par l'algorithme LRU, calculer le nombre de défauts de pages, et le coefficient de dégradation (défini dans la question précédente).

   Quelle serait la stratégie optimale, et le nombre de défauts de pages dans ce cas ?