Université Bordeaux I — U.F.R. Mathématiques et Informatique
Session de septembre 2000 — Alain Griffault

Maîtrise d'Informatique
Systèmes d'exploitation
(Durée : 3 heures)

Documents interdits

1 Cours : 2 points

Voici les premières lignes du fichier `dma.c` du noyau de Linux 2.0.35.

```
 1 /* $Id: dma.c,v 1.7 1994/12/28 03:35:33 root Exp root $
 2  * linux/kernel/dma.c: A DMA channel allocator. Inspired by linux/kernel/irq.c.
 3  *
 4  * Written by Hennus Bergman, 1992.
11  */
12
13 #include <linux/kernel.h>
14 #include <linux/errno.h>
15 #include <asm/dma.h>
16 #include <asm/system.h>
17
18
19 /* A note on resource allocation:
20  *
21  * All drivers needing DMA channels, should allocate and release them
22  * through the public routines 'request_dma()' and 'free_dma()'.
23  *
24  * In order to avoid problems, all processes should allocate resources in
25  * the same sequence and release them in the reverse order.
26  *
27  * So, when allocating DMAs and IRQs, first allocate the IRQ, then the DMA.
28  * When releasing them, first release the DMA, then release the IRQ.
29  * If you don't, you may cause allocation requests to fail unnecessarily.
30  * This doesn't really matter now, but it will once we get real semaphores
31  * in the kernel.
32  */
```

Question 1.1 *Expliquez les commentaires, sur l'allocation de ressources, compris entre les lignes 19 et 32.*

2 Producteur/Consommateur avec sémaphores : 5 points

Un processus `Producteur` dépose dans un buffer de taille `n` des messages qui sont ensuite retirés par un processus `Consommateur`.

Question 2.1 *Donnez le code des processus, et pour chacun des sémaphores utilisés, leur rôle, et leur valeur initiale.*

Le système comprend maintenant `m` couples `(Producteur(i), Consommateur(i))` et le buffer commun à tous est toujours de taille `n`, mais le processus `Consommateur(i)` ne peut pas retirer plus de messages qu'en a déposé le processus `Producteur(i)`.

Question 2.2 *Donnez le code des processus* `(Producteur(i), Consommateur(i))`*, et pour chacun des sémaphores utilisés, leur rôle, et leur valeur initiale.*

3 Allocation de ressources et interblocage : 4 points

Trois processus partagent trois ressources distinctes `R1, R2, R3`, avec accès exclusif et sans réquisition. Les processus disposent des procédures `allouer(ListeDeRessources)` et `liberer(ListeDeRessources)`, et chaque processus n'utilise pas plus de deux ressources.

Question 3.1 *Donnez un code pour chacun des processus de telle sorte qu'il existe un risque d'interblocage global des trois processus.*

Question 3.2 *Donnez un code pour chacun des processus de telle sorte qu'il existe un risque de privation (famine) pour un des trois processus, sans qu'il y ait possibilité d'interblocage.*

4 Banquier et interblocage : 3 points

On considère un système à une classe de ressources, où les ressources sont allouées et libérées une par une. Soit `n` le nombre de processus et `m` le nombre de ressources. L'annonce maximale pour chaque processus est au plus égale à `m` et la somme de toutes les annonces est inférieure strictement à `m+n`.

Question 4.1 *En vous inspirant de l'algorithme du banquier, (ou par démonstration directe) et en considérant un état où toutes les ressources sont prises, démontrez que l'interblocage est impossible.*

5 Gestion mémoire : 6 points

Soit une chaîne de références de pages pour un processus avec `m` cadres de pages (au début tous vides). La longueur de la chaîne de références est `p` avec `n` numéros de pages différents.

Question 5.1 *Donnez une limite inférieure sur le nombre de défauts de pages valable quelque soit l'algorithme de remplacement utilisé.*

Question 5.2 *Donnez une limite supérieure sur le nombre de défauts de pages valable quelque soit l'algorithme de remplacement utilisé.*

Question 5.3 *Vous avez inventé un nouvel algorithme de remplacement de pages, que vous pensez optimal. Votre algorithme présente dans quelques rares cas, l'anomalie de Belady. Votre algorithme est-il optimal ? Justifiez.*

Considérez le tableau à deux dimensions (le code est en pascal) :

```
var A : array[1..100] of array[1..100] of integer;
```

Les pages de la mémoire paginée peuvent contenir 200 entiers. Un processus occupe la page 0 (adresses de 0 à 199), la page 1 est occupée par les variables `A[1][1], .., A[1][100], A[2][1], .., A[2][100]` et ainsi de suite jusqu'à la page 50.
Le système comporte 3 cadres de pages, utilise l'algorithme LRU, et initialement, le premier cadre contient le processus (la page 0) alors que les deux autres sont vides.

Question 5.4 *Calculez le nombre de défauts de pages pour le processus suivant :*

```
for j := 1 to 100 do
     for i := 1 to 100 faire
          A[i,j] := 0;
```

Question 5.5 *Calculez le nombre de défauts de pages pour le processus suivant :*

```
for i := 1 to 100 do
     for j := 1 to 100 faire
          A[i,j] := 0;
```

Question 5.6 *Quelles sont les conséquences pour un programmeur de vos réponses aux deux questions précédentes ?*