Systèmes d'exploitation centralisés

1re année Informatique et Réseaux

17 juin 2015

Documents autorisés.

I Principes généraux (5 points)

1. Un processus consomme des ressources pour s'exécuter. Citer deux types de ressources qui lui sont indispensables.
2. Sur un système biprocesseur, est-il concevable qu'un processus donné puisse s'exécuter *simultanément* sur les deux processeurs ?
3. Système de fichiers : quel schéma d'utilisation de fichier est particulièrement bien adapté à l'allocation chaînée des blocs ?
4. La lecture sur disque se fait par bloc de taille fixe (par exemple 4 ou 8 Koctets). Cela est-il un inconvénient lorsqu'un programme lit séquentiellement dans un fichier par paquets de quelques octets ? Quel est le principe qui permet cela ?
5. Quel(s) mécanisme(s) permet(tent) d'assurer qu'un programme exécuté par un processus ne peut pas lire ou écrire les données d'un autre processus ?

II Noyau – Coroutines/processus (5 points)

Soit le code suivant :

```
#include <stdio.h>
#include "coroutines.h"
coroutine_t c1, c2, c3, c4;

void code1 (void *unused)
{
    printf("1\n");
    cor_transferer(c1,c3);
    printf("2\n");
    cor_transferer(c4,c4);
    printf("3\n");
}

void code2 (void *unused)
{
    printf("5\n");
    cor_transferer(c3,c4);
    printf("6\n");
}

void code3 (void *unused)
{
    printf("7\n");
    cor_transferer(c4,c1);
    printf("8\n");
    cor_transferer(c4,c2);
    printf("9\n");
}

int main()
{
    c1 = cor_creer("C1", code1, NULL);
    c2 = cor_creer("C2", code2, NULL);
    c3 = cor_creer("C3", code3, NULL);
    c4 = cor_creer("C4", NULL, NULL);
    cor_transferer(c4,c1);
    printf("10\n");
}
```

1. Détailler sommairement l'exécution du programme et indiquer ce qui s'affiche.
2. Donner un programme réalisant la même chose avec la couche processus. Le code doit garder la même structure, avec les appels à `cor_*` en moins, et des appels à `proc_*` en plus. Utiliser au choix toute combinaison de primitives de la couche processus (`activer`, `commuter`, `suspendre`, `continuer`...)
3. Préciser si un ordonnanceur spécifique est nécessaire (fifo, priorité...).

Note : du pseudo-C ou du langage algorithmique est bien suffisant, je ne suis pas un compilateur C pinailleur.

III Mémoire virtuelle (5 points)

Les questions portent spécifiquement sur la mémoire virtuelle implantée en TP, pas sur la mémoire virtuelle plus riche et plus complexe telle que présentée en cours.

La mémoire virtuelle implantée en TP utilise deux tables partagées (variables `infos_memcentrale` et `infos_swap`) et une table privée par processus (type `infos_proc`).

1. Expliquer comment est fait le lien entre un processus et sa table privée.
2. De la même manière qu'une page d'un processus suspendu peut être expulsée sur disque, on pourrait souhaiter expulser la table associée à un processus suspendu. Quelle(s) difficulté(s) cela pose-t-il ?
3. On souhaite enrichir le fonctionnement de la mémoire virtuelle implantée en TP pour que des processus puissent partager une page virtuelle (i.e. la page virtuelle des processus partageurs correspondra au même contenu). Pour cela, on ajoute une interface :

   ```
   void MV_share_page(void *adresse, processus_t avecqui);
   ```

   qui permet au processus courant d'indiquer que la page contenant l'adresse spécifiée est dorénavant partagée avec le processus `avecqui`.

   Préciser quelles sont les modifications à apporter à l'implantation actuelle, dans le cas où les appels à `MV_share_page` doivent nécessairement avoir lieu à l'initialisation, avant le moindre accès à la mémoire commune.
4. Même question, quand les appels à `MV_share_page` peuvent avoir lieu ultérieurement, alors que des processus ont déjà effectué des accès.

IV Systèmes de fichiers (5 points)

On souhaite étendre le système de fichiers UFS vu en cours en permettant d'utiliser deux tailles de blocs logiques : d'une part des petits blocs (8 ko), d'autre part des gros blocs (1 Mo). Un fichier n'est constitué que d'un seul type de blocs.

1. Quel est l'intérêt ?
2. À quel moment faut-il fixer la taille de blocs d'un fichier, et avec quels critères ?
3. Pour chacune des couches suivantes, justifier s'il est nécessaire ou pas de la modifier, et si oui comment :
   (a) Buffers
   (b) Inodes
   (c) Allocation des blocs
   (d) Accès données
   (e) Nommage