Système d'Exploitation
- le 17 Mai 2016, Durée : 2 heures

Toute réponse doit être justifiée
Machines à mémoire et Documents de Cours Autorisés

Exercice 1 *Questions de Cours*

1. Expliquez brièvement le mécanisme des appels système (vous pouvez choisir un exemple concret). Pour quelle raison un programme ne peut pas contourner le mécanisme et accéder directement au noyau ? **(1 pt)**
2. Pourquoi n'est-il pas envisageable, dans un système multi-utilisateurs, d'utiliser de simples appels procéduraux à la place du mécanisme des appels système ? **(1 pt)**
3. Certains appels système allouent des objets et retournent un identifiant de l'objet créé (souvent encapsulé dans un type opaque, comme par exemple les sémaphores). Pour quelle raison il serait dangereux de retourner un pointeur vers l'objet créé ? **(1 pt)**
4. Rappelez la différence entre fichiers de type `bloc` et fichiers de type `caractère`, et donnez pour chaque type un exemple de fichiers. **(1 pt)**
5. Sous `UNIX` le système gère trois tables pour gérer les fichiers, deux du côté noyau et un du côté processus utilisateur. Quelles sont ces tables et quels sont les avantages d'une telle implantation ? **(1 pt)**

Exercice 2 *Mémoire Virtuelle*

Un ordinateur dispose d'adresses codées sur 32 bits. Les adresses virtuelles sont composées de 8 bits pour la table des pages de niveau 1, de 12 bits pour la table des pages de niveau 2 et de 12 bits pour le déplacement dans la page.

1. Qu'est-ce qu'une adresse virtuelle ? **(1 pt)**
2. Faites un dessin illustrant la traduction d'une adresse virtuelle. Quelle est la taille des pages physiques ? **(1 pt)**
3. Combien de pages virtuelles totales existent ? **(1 pt)**

Exercice 3 *Synchronisation*

Un étudiant implante une simulation de trafic automobile à une intersection (entre deux voies) munie de feux tricolores. Cette simulation s'appuie sur deux programmes `UNIX`.

— Le programme carrefour exécuté par un seul processus et qui a pour tâche de gérer les feux tricolores. Il est lancé au début de la simulation.
— Le programme voiture est exécuté par les autres processus en parallèle.

Tous les programmes ont accés à une zone mémoire partagée qui contient les variables suivantes :

```
int peux_passer = 1; /* un bouléen : 1 ou 0 */
int sens_actif = 1; /* 1 ou 2 */
int voitures_engagees = 0; /* le nombre de voitures en train de
                              traverser le carrefour */
```

Voici le code du programme carrefour.

```
while (1) {
  sleep (PERIODE);
  peux_passer  = 0;
  while (voitures_engages > 0) /* ne rien faire */
    ;
  sens_actif = 3 - sens_actif; /* l'autre sens peut passer */
  peux_passer = 1;
}
```

Voici le code du programme voiture.

```
int sens = sens_random ();
while (!peux_passer || sens != sens_actif) /* attendre */
  ;
voitures_engagees++;
sleep (TRAVERSEES);
voitures_engagees--;
```

1. Supposons que l'on dispose de n processus voitures. Expliquez comment lancer, depuis le programme principal, les n processus voitures plus le processus carrefour sans créer de processus zombies lorsqu'un des $n + 1$ processus est fermé. **(1 pt)**
2. Expliquez les 2 programmes proposés par l'étudiant. Vous expliquerez en particulier le rôle des différentes variables (5 phrases maximum par programme). **(2 pts)**
3. Pour quelles raisons les boucles while dans les deux programmes font que les processus sollicitent intensivement le(s) processeur(s). **(1 pt)**
4. Quelles sont les sections critiques de la simulation ? **(1 pt)**
5. Proposez un scénario qui provoque ou (ou des) accident(s), *i.e.*, plusieurs voitures traversant en même temps. **(1 pt)**
6. Dans le cours nous avons vu trois méthodes pour gérer des sections critiques : le masquage des interruptions, la méthode du TSL, et les sémaphores. Décrivez le principe de fonctionnement pour chacun d'eux. Vous donnerez également un tableau comparatif des avantages et inconvénients des trois. **(3 pts)**
7. Corrigez le protocole de la simulation à l'aide des sémaphores. N'oubliez pas de donner les valeurs d'initialisation des sémaphores. **(2 pts)**
8. On se propose d'ajouter à la simulation des véhicules prioritaires (pour gérer par exemple les voitures d'urgence comme le samu ou les pompiers). Le programme gérant ces véhicules est appelé voiture-prioritaire. Ces véhicules sont toujours prioritaires

aux intersections. Lorsqu'un tel véhicule arrive à une intersection, il dissuade les autres véhicules d'entreprendre la traversée du carrefour, attend que les voitures déjà engagées terminent la traversée et enfin traverse sans s'occuper de la couleur du feu.

Étendez le protocole précédent et donnez le code du programme voiture-prioritaire de manière à respecter les comportements décrits ci-dessus. On suppose que deux véhicules prioritaires ne se rencontrent jamais à un carrefour. **(3 pts)**