

INF3172 — Principes des systèmes d'exploitation Examen Final — Automne 2016

Samedi 17 décembre — Durée 2 heures

<input type="checkbox"/> 0							
<input type="checkbox"/> 1							
<input type="checkbox"/> 2							
<input type="checkbox"/> 3							
<input type="checkbox"/> 4							
<input type="checkbox"/> 5							
<input type="checkbox"/> 6							
<input type="checkbox"/> 7							
<input type="checkbox"/> 8							
<input type="checkbox"/> 9							

← Codez les 8 chiffres de votre code permanent ci-contre, et inscrivez-le à nouveau ci-dessous avec vos nom et prénom.

Code permanent :
.....
Nom :
.....
Prénom :
.....

Aucun document n'est autorisé. L'usage de la calculatrice ou de tout équipement électronique est interdit. Les sacs, étuis et autre effets personnels doivent être fermés et rangés. Toute sortie de la salle d'examen est définitive.

Les questions faisant apparaître le symbole ♣ peuvent présenter plusieurs bonnes réponses. Les autres ont une unique bonne réponse.

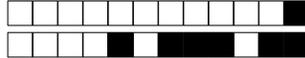
Des points négatifs pourront être affectés à de très mauvaises réponses.

Important : noircissez complètement l'intérieur de chaque case (pas de croix, pas de cercles).

1 Questions à choix multiples (60 points)

Question 1 ♣ Parmi les affirmations suivantes sur la communication par signaux, indiquez toutes celles qui sont vraies. (15 points)

- Un processus peut demander d'ignorer certains signaux.
- Les signaux permettent à un utilisateur d'envoyer de l'information à un autre utilisateur.
- Lors d'un `exec`, les gérantes de signaux (*signal handlers*) sont préservées.
- Lors d'un `fork`, les gérantes de signaux (*signal handlers*) sont préservées.
- Lorsque l'appel système `kill` retourne 0, le processus auquel il est envoyé s'est nécessairement terminé.
- Aucune de ces réponses n'est correcte.



Question 2 ♣ Parmi les affirmations suivantes sur la communication par tubes (pipe), indiquez toutes celles qui sont vraies. (15 points)

- Tous les appels système qui utilisent des descripteurs de fichiers fonctionnent autant sur les tubes que sur les fichiers réguliers.
- Un tube a une capacité limitée.
- L'appel système `pipe` permet de se connecter à un tube existant.
- Un tube est un canal de communication interprocessus unidirectionnel.
- Lorsque l'on programme avec des tubes, fermer les descripteurs de fichiers superflus (`close`) sert seulement à économiser des descripteurs de fichiers et de la mémoire.
- Par défaut, l'appel système `read` bloque s'il est utilisé sur un tube vide mais qui a des écrivains.
- Aucune de ces réponses n'est correcte.*

Question 3 ♣ Parmi les affirmations suivantes sur les sections critiques, indiquez toutes celles qui sont vraies. (15 points)

- Seul un processus prioritaire peut entrer en section critique.
- Il y a, au maximum, une seule section critique par programme.
- Un processus dans une section critique peut être interrompu par une interruption matérielle.
- Une section critique protège au maximum une seule variable partagée.
- Un processus dans une section critique reste à l'état actif.
- Au maximum, un seul processus peut être à la fois en section critique.
- Aucune de ces réponses n'est correcte.*

Question 4 ♣ Parmi les affirmations suivantes sur la gestion de la mémoire, indiquez toutes celles qui sont vraies. (15 points)

- Lors d'un défaut de page, le système d'exploitation doit ramener la page manquante à partir du disque.
- Le MMU du processeur traduit les adresses virtuelles en adresses physiques (réelles).
- Le système d'exploitation est invoqué à chaque accès mémoire afin de l'autoriser ou non.
- La mémoire virtuelle permet l'exécution de programmes dont la taille excède la mémoire réelle.
- Les registres base et limite permettent de reloger un processus ailleurs en mémoire.
- Lorsque la mémoire est pleine, le système d'exploitation peut en télécharger davantage de l'internet.
- Aucune de ces réponses n'est correcte.*



2 Questions à développement (40 points)

Ces questions mettent en uvre différents concepts des systèmes d'exploitation.

Les cas d'utilisation étudiés se font dans le contexte du TP2 sur les files de commandes. Il n'est toutefois pas nécessaire d'avoir réussi le TP pour pouvoir répondre aux questions.

Les questions de cette partie sont relatives aux deux commandes suivantes (qui affichent `hello` à l'écran):

```
$ cq -s &
$ cq -c echo hello
```

La commande `ncq -s` crée un serveur de file de commandes. Pour ce faire, elle crée un socket UNIX `cq.socket` et attend les clients. Les clients qui se connectent sont mis dans une file d'attente et l'autorisation de sexécuter leur est envoyé dans l'ordre d'arrivée, un seul client à la fois.

La commande `ncq -c echo hello` se connecte à ce serveur, attend le feu vert du serveur, exécute la commande `echo hello` dans un sous-processus, attend la fin du sous-processus, puis se termine en fermant le socket.

Pour rappel, le protocole entre les clients et le serveur est le suivant:

- quand le client se connecte au serveur, il lui envoie le caractère "R" (*ready/red*) pour signaler qu'il veut se mettre dans la file d'attente.
- quand c'est le tour d'un client (le premier dans la file), le serveur lui envoie le caractère "G" (*go/green*) pour signaler qu'il peut s'exécuter. Le client devient actif et exécute la commande.
- quand le client actif termine, il ferme la connexion avec le serveur pour signifier la fin de l'exécution. C'est alors le tour du client suivant dans la file.

2.1 Appels système

Soit la liste d'appels système suivants

- `int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);` Accepter une connexion sur un socket.
- `int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);` Fournir un nom public à un socket.
- `int close(int fd);` Fermer un descripteur de fichier.
- `int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);` Débuter une connexion sur un socket.
- `int execve(const char *filename, char *const argv[], char *const envp[]);` Exécuter un programme.
- `pid_t fork(void);` Créer un processus fils.
- `ssize_t read(int fd, void *buf, size_t count);` Lire depuis un descripteur de fichier.
- `ssize_t recv(int fd, void *buf, size_t count, int flags);` Recevoir un message sur un socket.
- `ssize_t send(int fd, const void *buf, size_t count, int flags);` Envoyer un message sur un socket.
- `int socket(int domain, int type, int protocol);` Créer un point de communication.
- `pid_t wait(int *status);` Attendre la fin dun processus.
- `ssize_t write(int fd, const void *buf, size_t count);` Écrire dans un descripteur de fichier.

Question 5 Indiquez la séquence de la demi-douzaine d'appels système (parmi ceux de la liste) utilisés par le processus principal de `ncq -c echo hello` (le processus fils est traité dans la question suivante). (5 points)

Note: n'écrivez pas de programme C. Indiquez juste la liste dans l'ordre des appels système exécutés par le processus principal jusqu'à sa terminaison.

Attention: pour chaque appel système, indiquez lorsque c'est pertinent les arguments et la valeur de retour.

0 5

.....



Question 6 Indiquez la séquence complète des appels système utilisés par le sous-processus (celui qui exécute le echo) jusqu'à sa terminaison. (5 points)

0 5

.....

2.2 Processus et descripteurs de fichiers

Question 7 En comptant le shell, les cq et le echo, indiquez combien de processus sont impliqués en tout. (2 points)

1 2 3 4 5 6 7 8

Question 8 Faites un dessin illustrant la relation de parenté entre chacun des processus impliqués. (3 points)

0 3

.....

Question 9 Pour chacun des processus impliqués identifiez tous ses descripteurs de fichiers et expliquez à quoi correspond chacun d'eux. (5 points)

0 5

.....

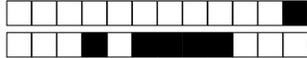
2.3 Cas limites

Question 10 Sur les mêmes deux commandes, expliquez en détail les raisons pour lesquelles il est possible que la commande `ncq -c echo helloz` affiche un message d'erreur comme quoi le fichier socket `cq.socket` n'existe pas. (10 points)

Note: on considère que (a) le serveur et le client n'ont pas de bogues; (b) le serveur est capable de se lancer et de créer correctement la socket et le fichier `cq.socket` associé; et (c) qu'aucun processus tiers n'intervient directement ou indirectement.

0 5 10

.....



Question 11 L'utilisateur décide de compliquer les choses et écrit

```
$ cq -c cq -c echo goodbye
```

Qu'affiche cette ligne de commande ? Justifiez en détail votre raisonnement. (10 points)

0 5 10

.....