Nom :

Prénom :

Code permanent :

Place :

INF3172 — Principes des systèmes d'exploitation

Examen Intra

Mardi 9 mars 2009 — Durée 2 heures

Notes :
— Tout document interdit.
— Les réponses aux questions doivent être argumentées et aussi concises que possible.
— Le barème est donné à titre indicatif.
— La lisibilité et la qualité du code que vous écrivez est pris en compte dans la notation.

1 Modes d'exécution (15 points)

Des instructions machine suivantes, indiquez celles qui sont utilisables en mode utilisateur et celles qui nécessitent le mode noyau. Justifiez brièvement votre réponse.

1. Inactivation des interruptions matérielles

   .....

2. Invocation d'une fonction C située à une adresse donnée

   .....

3. Instruction TRAP effectuant un appel système

   .....

4. Changement de l'heure de l'ordinateur

   .....

5. Addition de deux registres CPU généraux

   .....

2 État des processus (15 points)

1. Un processus p_1 fait une demande d'entrée-sortie au système d'exploitation. Quels sont les états possibles de ce processus ? Justifiez votre réponse.

   .....

   .....

   .....

2. Un processus p_2 fait un `sleep(5)` (appel système pour s'endormir 5 secondes). Quels sont les états possibles de ce processus ? Justifiez votre réponse.

   .....

   .....

   .....

3. Un processus p_3 fait un `exit(0)`. Quels sont les états possibles de ce processus ? Justifiez votre réponse.

   .....

   .....

   .....

3 Segments de mémoire (15 points)

1. Dans quel segment mémoire sont stockées les variables locales et quelle est leur durée de vie ?

   .....

   .....

2. Dans quel segment mémoire sont stockées les données allouées par `malloc` et quelle est leur durée de vie ?

   .....

   .....

3. Dans quel segment mémoire est stocké le code machine des fonctions du programme et quelle est sa durée de vie ?

   .....

   .....

4 Génération et recouvrement de processus (25 points)

1. Expliquez le rôle de l'appel système `fork()`.

   .....

   .....

   .....

   .....

   .....

   .....

2. L'appel système `exec()` peur échouer pour des tas de raisons. Listez et expliquez trois de ces raisons.

   .....

   .....

   .....

   .....

   .....

   .....

5 Système de fichiers et déarchiveur goudron(30 points)

1. Soit la séquence de commandes shell suivante :

```
$ echo "world" > hello
$ ls -li hello           #0
$ ln hello goodby
$ chmod 600 hello
$ ls -li hello goodby    #1
$ cat goodby             #2
$ rm hello
$ ls -li goodby          #3
```

   Sachant que la commande `#0` affiche

```
  6833165 -rw-r--r-- 1 privat privat 6  1 mar 12:22 hello
```

   indiquez ce qu'affichent les commandes `#1`, `#2` et `#3`.

   Les questions suivantes sont relatives au TP 1 sur l'outil `goudron`.

2. Normalement votre désarchiveur goudron ne doit pas écrire en dehors du répertoire de destination. En effet, il était demandé de détecter les noms de fichiers contenant `..` ou commençant par `/`. Toutefois, cette mesure de sécurité n'est pas suffisante.

   Montrez comment il est possible d'abuser des liens symboliques afin de générer une archive valide qui extrait (ou pire écrase) des fichiers en dehors du répertoire de destination.

   .....

   .....

   .....

   .....

   .....

3. Soit une archive au format goudron contenant :
   — un répertoire `dru` avec les droits `r-x------` ;
   — un ficher régulier `dru/flop` avec les droits `r--r--r--` et de contenu `coucou` (de taille 6 octets donc) ;
   — un lien symbolique `hyrule` qui pointe vers le fichier `dru/flop`.

   Écrivez la séquence des appels systèmes que doit exécuter le désarchiveur `goudron` afin d'extraire cette archive dans le répertoire `dest`. N'écrivez qu'un seul appel par ligne en précisant la valeur des arguments et en indiquant la valeur de retour selon la forme

   `appelSysteme(valeurArgument1, valeurArgument2, etc.)` $\rightarrow$ `valeurRetour`

   On ne s'intéresse aux appels systèmes suivants (tous ne sont pas nécessaires) :
   — `int close(int fd)` - Fermer un descripteur de fichier
   — `int creat(const char *pathname, mode_t mode)` - Créer un fichier
   — `int chmod(const char *path, mode_t mode)` - Modifier les permissions d'accès à un fichier
   — `int mkdir(const char *pathname, mode_t mode)` - Créer un répertoire
   — `int open(const char *pathname, int flags)` - Ouvrir un fichier
   — `DIR *opendir(const char *nom)` - Ouvrir un répertoire
   — `ssize_t read(int fd, void *buf, size_t count)` - Lire depuis un descripteur de fichier
   — `int symlink(const char *oldpath, const char *newpath)` - Créer un lien symbolique vers un fichier
   — `ssize_t write(int fd, const void *buf, size_t count);` - Écrire dans un descripteur de fichier