Département Informatique

Nom :	Année scolaire : 2004–2005
Prénom :	Date : 14 janvier 2005

Module ISI423
Session de janvier
—
Systèmes d'exploitation et leur support d'exécution
—
Contrôle de connaissance[1] de 45 minutes

Merci de répondre (au moins) dans les blancs.
Lire tout le sujet en entier du début à la fin, en commençant à la première page et jusqu'à la dernière page, avant de commencer à répondre : cela peut vous donner de l'inspiration et vous permettre de mieux allouer votre temps en fonction de vos compétences.
Chaque question sera notée entre 0 et 10 et la note globale sera calculée par une fonction des notes élémentaires. La fonction définitive sera choisie après correction des copies.
Attention : tout ce que vous écrirez sur cette copie pourra être retenu contre vous, voire avoir une influence sur la note d'ISI423.

1 Généralités système

Question 1 : Quel est l'intérêt d'utiliser une machine virtuelle par rapport à une machine physique ? À quoi cela peut servir ? Quels sont les inconvénients ?

→
→
→
→

[1] Avec document, avec calculatrice, sans triche, sans copie sur les voisins, sans micro-ordinateur portable ou non, sans macro-ordinateur, sans téléphone portable ou non, sans oreillette de téléphone ni de dictaphone, sans talkie-walkie, sans télépathie, sans métempsycose, sans pompe, avec anti-sèche, avec tatouage ou vêtement imprimé en rapport avec le sujet, avec mouchoir de poche pré-imprimé, avec piercing ou scarifi cation en rapport avec l'ISI423,...

→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→

Question 2 : Pourquoi doit-on généralement rajouter une structure de système de fichiers à un disque dur pour pouvoir l'utiliser ?

→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→

2 Temps partagé

Question 3 : Un système à temps partagé fait des changements de contextes à une fréquence de 1 kHz. Un changement de contexte prend 30 μs. Quelle proportion de processeur reste disponible pour faire fonctionner les programmes des utilisateurs ?

→
→
→
→
→

Question 4 : Quels sont les avantages et inconvénients liés à une augmentation ou une diminution de la fréquence des changements de contextes ?

→
→
→
→
→
→
→
→
→
→
→
→
→
→

Question 5 : Le programmeur du jeu sur ordinateur Matrix simule les interactions de 3 joueurs, Néo, Smith et Trinity sous forme de 3 processus légers (*threads*) de type noyau. Chaque processus met respectivement 10, 20 et 10 μs à s'exécuter. Un cycle de simulation consiste à exécuter successivement chaque processus pour calculer les interactions entre joueurs (salutation, coup de poing, gifle, passage par la fenêtre, etc.) et chaque changement de contexte prend là encore 20 μs. À quelle fréquence maximale les interactions du jeu sont-elles faites ? Sachant que plus il y a d'interactions et plus le jeu est fluide et physiquement réaliste, utiliser des processus légers de type utilisateur a-t-il un intérêt ?

→
→
→
→
→
→

→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→
→

Majeure ISI
UV2 – ISI-423 – Partie Programmation système
Contrôle de connaissances
14 janvier 2004

Question 1

Vous trouverez en annexe la page du manuel de référence concernant la fonction system(3) qui facilite le travail du programmeur désirant exécuter une commande Unix directement à partir de son programme.
Donnez votre version de l'implémentation de cette fonction. Dans une première partie vous indiquerez l'architecture générale (quelles fonctionnalités mettez vos en œuvre ?). Ensuite, dans une seconde partie, vous donnerez votre propre version du code (nous ne vous demandons pas d'être précis au point-virgule près).
Nota : nous n'avons pas bien vu en cours et en TP comment bloquer un signal, or il est question de cela ici, entre-autre. Si vous savez comment faire, indiquez le, sinon contentez-vous de mentionner l'action (au bon endroit, évidemment).

Annexe

SYSTEM(3) Linux Programmer's Manual SYSTEM(3)

NAME

system - execute a shell command

SYNOPSIS

#include <stdlib.h>

int system(const char *string);

DESCRIPTION

system() executes a command specified in string by calling **/bin/sh -c** string, and returns after the command has been completed. During execution of the command, **SIGCHLD** will be blocked, and **SIGINT** and **SIGQUIT** will be ignored.

RETURN VALUE

The value returned is -1 on error (e.g. fork failed), and the return status of the command otherwise. This latter return status is in the format specified in **wait(2)**. Thus, the exit code of the command will be WEXITSTATUS(status). In case **/bin/sh** could not be executed, the exit status will be that of a command that does exit(127).

If the value of string is **NULL**, **system()** returns nonzero if the shell is available, and zero if not.

system() does not affect the wait status of any other children.

CONFORMING TO

ANSI C, POSIX.2, BSD 4.3

NOTES

As mentioned, **system()** ignores SIGINT and SIGQUIT. This may make programs that call it from a loop uninterruptable, unless they take care themselves to check the exit status of the child. E.g.

```
while(something) {
    int ret = system("foo");

    if (WIFSIGNALED(ret) &&
        (WTERMSIG(ret) == SIGINT || WTERMSIG(ret) == SIGQUIT))
            break;
}
```

Do not use **system()** from a program with suid or sgid privileges, because strange values for some environment variables might be used to subvert system integrity. Use the **exec**(3) family of functions instead, but not **execlp**(3) or **execvp**(3). **system()** will not, in fact, work properly from programs with suid or sgid privileges on systems on which **/bin/sh** is bash version 2, since bash 2 drops privileges on startup. (Debian uses a modified bash which does not do this when invoked as sh.)

The check for the availability of /bin/sh is not actually performed; it is always assumed to be available. ISO C specifies the check, but POSIX.2 specifies that the return shall always be non-zero, since a system without the shell is not conforming, and it is this that is implemented.

It is possible for the shell command to return 127, so that code is not a sure indication that the **execve()** call failed.

SEE ALSO

sh(1), **signal**(2), **wait**(2), **exec**(3)