L3 Informatique
Université de Bourgogne

U.E. Systèmes & Réseaux I

Exercices communications inter-processus

1 Signaux

Exercice 1- Écrire deux programmes shell, l'un affichant ping l'autre affichant pong de manière à ce qu'ils se coordonnent pour afficher ping-pong-ping...

Exercice 2- Écrire un programme shell qui refuse pendant une phase critique de son exécution les interruptions clavier.

Exercice 3- *Utilisation en boucle de deux fichiers*

Le système à modéliser est constitué d'un processus principal P_1 qui demande à l'utilisateur des commandes, les ajoute à un fichier de journalisation et les exécute. Un second processus P_2 va se charger de l'exploitation du fichier de journalisation (pour éditer des statistiques par exemple).

1. Quel est le problème à régler ? En quoi l'utilisation de deux fichiers permet-elle de simplifier le problème ?

2. On suppose tout d'abord que le temps d'exécution de P_2 est négligeable par rapport à celui de P_1. La structure choisie est la suivante : P_1 déclenche la bascule d'un fichier à l'autre lorsqu'un nombre précis de commandes est archivé. Il envoie alors un signal a P_2. P_2 est en boucle d'attente et en sort à l'arrivée du signal.

3. On suppose que P_2 est presque toujours plus court que P_1 et que la taille maximum du fichier de journalisation n'est pas impérative.

4. Idem mais en supposant que la taille maximum de fichier doit être strictement respectée.

2 Tubes

Exercice 4- Écrire un programme C qui crée un tube puis se duplique. Le père envoie deux chaînes de caractères dans le tube (Hello puis World) qui sont lues par le fils (FIFO). Le père coupe ensuite l'entrée du tube ce qui a pour effet de renvoyer zéro au du fils et de le faire sortir. Le père se synchronise puis se termine.

Exercice 5- Reprendre le programme précédent, créer un tube nommé et deux processus n'ayant pas de liens de filiation.

Exercice 6- Écrire un programme qui copie un fichier ou plus généralement l'entrée (standard) sur la sortie standard en utilisant les fonctions read et write.

Exercice 7- Réaliser un programme qui indique la taille et la nature des fichiers passés en paramètre.
Réaliser un programme qui prend en argument un fichier et le passe à la commande unix `wc`.

3 Sémaphores, exclusion

Exercice 8- *Verrouillage*

La fonction fcntl() offre la possibilité de ne bloquer que des portions limitées d'un fichier. Écrire un programme qui demande un verrouillage en écriture des fichiers dont les noms lui sont passés en argument et attend que l'utilisateur ait pressé la touche Entrée avant de se terminer (libérant ainsi les verrous).

Exercice 9- *Ordonnancement*

Soient T1, T2, T3, T4, T5 et T6 des tâches qui doivent être exécutées selon l'ordre suivant:
- T1 est la tâche initiale
- T2, T3, et T4 ne commencent que lorsque T1 est terminée
- T5 ne commence que lorsque T2 et T3 sont terminées
- T6 ne commence que lorsque T4 et T5 sont terminées

Résoudre ce problème d'ordonnancement en utilisant des sémaphores.

Exercice 10- *Pendu en concurrence*

On veut faire jouer au pendu plusieurs processus. Chaque processus joueur, lorsqu'il a la main, propose une lettre au processus maître du jeu. Si la lettre est correcte, le maître du jeu la rend visible dans le mot à chercher. On suppose que:
- Jouer est protégé par un sémaphore
- Tout joueur qui a la main va 1) proposer une lettre, 2) voir l'état du mot modifié s'il a trouvé une lettre existant dans le mot ou avoir un message d'échec.
- Le mot sur lequel on joue est dans une zone de mémoire partagée.

Écrire les grandes lignes du programme du processus maître.

Annexe: Principaux signaux Unix

numéro	nom	événement
1	SIGUP	envoyé à tous les processus d'un terminal lors de la déconnexion
2	SIGINT	envoyé à tous les processus d un terminal lors de l'utilisation de la touche INTR (DEL)
3	SIGQUIT	envoyé à tous les processus lors de l'utilisation de la touche QUIT (CTR \)
4	SIGILL	envoyé en cas d'opération illégale
5	SIGTRAP	envoyé après chaque instruction en mode trace (pour les débogueurs)
8	SIGFPE	indicateur d'erreur lors d'une opération en virgule flottante
9	SIGKILL	destruction inconditionnelle du processus destinataire
11	SIGSEGV	envoyé en cas de violation de segment mémoire
12	SIGSYS	indicateur d'une erreur de paramètre dans un appel système
13	SIGPIPE	indicateur d'une erreur dans un pipe (écriture sans lecteur)
14	SIGALARM	signal d'horloge
15	SIGTERM	demande de terminaison d'un processus
16	SIGUSR1	rien : à disposition de l'utilisateur
17	SIGUSR2	rien : à disposition de l'utilisateur

Les signaux SIGKILL et SIGSTOP ne peuvent être ni capturés ni ignorés.