SR02 : TD 2

TD Machine sur une seule séance de 2h

Exercice 1. (Prélèvement périodique de température)

Nous désirons développer un programme qui permet de faire un prélèvement de température chaque 5 secondes. Entre deux mesures, le processus est occupé à faire autre chose (par exemple afficher une barre de progression (----------). Pour ce faire, on utilisera une horloge qui émet un signal SIGALRM après un délai spécifié avec la fonction alarm (....) ; On considère que le thermomètre est un processus fils créé avec fork(). Ce thermomètre reste endormi dans une boucle (pause ()), et pour mesurer la température il faudra le réveiller avec le signal SIGUSR1. Quand le thermomètre reçoit le signal SIGUSR1, il se réveille et pour simuler la mesure de température, ce dernier tire un nombre aléatoire, compris entre 10 et 40, avec la fonction random(), et l'affiche en tant que température.

- Programmez ce prélèvement périodique de température, en redéfinissant les handlers des signaux SIGUSR1 (pour le processus thermomètre) et SIGALRM (pour le processus père).

Indice : Afin de bien utiliser les fonctions alarm(), pause(), random(), fork(), kill(), sigaction(), etc., documentez-vous avec la commande man d'unix.

Exercice 2. (Synchronisation par signaux)

Créer un programme dans lequel un processus crée un processus fils. Les deux processus communiquent en utilisant le signal SIGUSR1. Le fils affiche des lettres minuscules de « a » à « z ». Le père affiche les lettres majuscules de « A » à « Z ».

- Ecrire le programme de telle sorte que le texte affiché sera comme suit :
  aAbcBCdefDEFghijGHIJklmnoKLMNOpqrstuPQRSTUvwxyzVWXYZ

Indice : installer un handler de SIGUSR1 (en utilisant sigaction) au niveau du père et un autre handler au niveau du fils. Quand un processus affiche sa part de lettres, il réveille l'autre processus en lui envoyant un signal kill(pid,SIGUSR1), puis il attend (pause()) qu'il soit réveillé par l'autre processus quand il aura terminé d'afficher sa part de lettres.

Exercice 3. (Simulation d'un système de gestion de tâches avec communication inter-processus)

Écrivez un programme en langage C sous Unix qui simule un système de gestion de tâches basique. Le programme doit inclure les fonctionnalités suivantes :

1. Le processus principal crée un processus fils, appelé "gestionnaire de tâches".
2. Le gestionnaire de tâches crée une file d'attente (par exemple, une pipe) pour recevoir des demandes de tâches provenant d'autres processus.
3. Le processus principal doit également installer un gestionnaire de signal pour capturer le signal SIGINT (Ctrl+C).
4. Lorsque le signal SIGINT est reçu par le processus principal, celui-ci doit envoyer un signal SIGUSR1 au gestionnaire de tâches pour lui indiquer de terminer son exécution.
5. Les autres processus (appelés "clients de tâches") peuvent envoyer des demandes de tâches au gestionnaire de tâches via la file d'attente.
6. Chaque demande de tâche doit être encapsulée dans une structure de données appropriée, contenant au moins l'identifiant de la tâche et ses paramètres.

7. Le gestionnaire de tâches doit traiter les demandes de tâches dans l'ordre d'arrivée et exécuter chaque tâche dans un processus enfant séparé (utilisation de fork).
8. Une fois qu'une tâche est terminée, le gestionnaire de tâches doit envoyer un message de retour au client de tâches approprié pour signaler l'achèvement de la tâche.
9. Le gestionnaire de tâches doit également gérer les cas où un processus enfant dépasse un certain temps d'exécution et le tuer si nécessaire, en envoyant un signal approprié au processus enfant.
10. Assurez-vous que le programme gère correctement la terminaison de tous les processus et nettoie les ressources allouées avant de se terminer.