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. (Application avec interface graphique)

- Écrire un programme sig.c qui va capter le signal SIGINT (Control-C). Ce programme sera linké avec les fonctions précompilées dans sigx.o (fichier à télécharger) par la commande :

> gcc -o sig sig.c sigx.o -L/usr/X11R6/lib -lX11

sig.c va successivement :

- capter le signal SIGINT dans une fonction "captpere"
- créer un process fils qui lui-même va :
  - créer une fenêtre graphique par appel à initrec()
  - capter le signal SIGINT dans une fonction "captfils"
  - boucler sur i = attendreclic() jusqu'à i =-1 (clic sur _fin_)
  - si le fils reçoit un clic dans le bouton "_0_", il envoie un signal SIGINT à son père
  - si le fils reçoit un signal SIGINT, il mets son rectangle en vert pendant 5 secondes.
- de plus :
  - la routine "captpere" va compter le nombre de SIGINT, et provoquer la sortie du process père au troisième SIGINT

- la routine "captfils" va compter le nombre de SIGINT, et détruire la fenêtre graphique et tuer le process fils au troisième SIGINT
- le père s'endort en appelant dans une boucle "n=sleep (10);" et affiche le temps d'attente qui restait à faire lorsque cet appel a été interrompu par un SIGINT (control-C).

•
○

Indice

Vous pouvez exécuter le programme suivant pour avoir une idée du comportement demandé (fichier à télécharger).

Les fonctions à utiliser dans sigx.o

- initrec() : initialise fenêtre rectangle rouge et "boutons"

- i = attendreclic(): boucle attente événements
  - sort avec i =-1 si clic dans rect-fin
  - sort avec i = 0 si clic dans grand rect. "_0_"
  - sort avec i = 1 si clic dans grand rect. "_1_"
  - etc ...
- rectvert(n) : met le rectangle en vert pendant n secondes, puis le remet a sa couleur précédente.
- detruitrec(): détruit la fenêtre rectangle
- ecritrec (char *buf, int lng) : écrire buf dans la fenêtre