TP 3 - PSE

2 séances (MAX.) en salle Machine

Les signaux classiques

Objectifs :

Les objectifs de ce TP sont de se familiariser avec les signaux et de maitriser les *appels Système* qui permettent de gérer les signaux : ***strsignal***, ***kill***, ***sigemptyset***, ***sigfillset***, ***sigdelset***, ***sigprocmask***, ***sigpending***, ***signal***, ***sigaction***.

Travail à effectuer :

1) Nous allons écrire un programme C qui permet de consulter les libellés des signaux.

a) Tout d'abord, pensez à lire le cours et à consulter le manuel Linux (la commande ***man***)
b) N'oubliez pas d'inclure les librairies nécessaires
c) Utilisez par exemple la fonction ***strsignal()*** et/ou la chaine de caractères ***sys_siglist[]***
d) Demandez éventuellement l'aide de votre chargé de TP.

Pour information :
La fonction ***strsignal****( )* est disponible en tant qu'extension GNU, il faut donc utiliser la constante symbolique _GNU_SOURCE à la compilation dans <string.h>.
Il existe une table globale de chaînes de caractères contenant les libellés des signaux : *char ** ***sys_siglist****[numéro_signal]*.

2) Nous allons écrire un programme C pour pouvoir installer notre premier gestionnaire de signal et tenter de capturer tous les signaux.

a) Tout d'abord, pensez à lire le cours (diapo. 112-116) et à consulter le manuel Linux (la commande ***man***)
b) N'oubliez pas d'inclure les librairies nécessaires

c) Utilisez pour cette question la fonction ***signal()***
d) Demandez éventuellement l'aide de votre chargé de TP.

Pour information :
La fonction ***signal()*** échouera à capturer **SIGKILL** et **SIGSTOP**.
Le programme devra afficher le PID du processus en cours, suivi du numéro de signal et de son nom.
Il faudra disposer d'une seconde console pour pouvoir envoyer des signaux et pour tuer le processus à la fin (Ctrl+c).
Pour envoyer des signaux : nous pouvons utiliser les touches Ctrl+c (Interrupt), Ctrl+z (Stopped) et Ctrl+Alt Gr+\ (Quit) sur la même console du processus en exécution **ou** à travers des ordres kill envoyés depuis une autre console ; exemple : ***kill –TERM PID_PROCESSUS***.

3) Il existe toutefois de nombreux systèmes Unix (de la famille Système V) sur lesquels un gestionnaire de signal ne reste pas en place après avoir été invoqué (une fois que le signal est arrivé, le noyau repositionne le comportement par défaut).
Ce dernier comportement peut être observé sous Linux avec GlibC en définissant la constante symbolique _XOPEN_SOURCE avant d'inclure <signal.h>. ***Votre chargé de TP va vous montrer un exemple sur ce comportement.***

4) Nous allons pouvoir faire la même chose que la question précédente en utilisant cette fois ci l'appel système ***sigaction()***, donc :
a) Tout d'abord, pensez à lire le cours et à consulter le manuel Linux (la commande ***man***)
b) N'oubliez pas d'inclure les librairies nécessaires
c) Utilisez bien évidemment la fonction ***sigaction()***
d) Justifiez dans votre compte rendu les initialisations que vous avez faites sur les arguments de la fonction ***sigaction()***
e) Demandez éventuellement l'aide de votre chargé de TP.

Pour information :
La fonction ***sigaction*** *()* échouera elle aussi à capturer **SIGKILL** et **SIGSTOP**.
Il faudra initialiser la(es) structure(s) sigaction (sa_handler, sa_mask et sa_flags) avant l'exécution de l'appel système *sigaction*

5) Nous allons maintenant voir comment utiliser les appels système ***sigprocmask***, ***sigpending*** et autres pour bloquer/débloquer et mettre en attente des signaux. Pour cela, nous allons étudier et tester le programme ***5-function-sigprocmask-sigpending.c***

6) Vous allez maintenant vous baser sur les fonctionnalités vues en questions 1) 2) 3) 4) et 5) pour faire fonctionner le programme que vous avez vu en cours (travail personnel).