1516 L2 SEPS TP 4 - Systèmes d'Exploitation et Programmation Système

Signaux Unix

Ce TP se fait sur les machines Unix. Vous devez rendre un rapport et vos programmes à la fin du cours sur Espadon. Assurez-vous que votre rapport porte un nom du type "1516_L2_SEPS_TDTP_4_Signaux_Unix_*votreNom*". Mettez dans le rapport des copies d'écran des résultats d'exécutions ("Applications" → "Accessoires" → "Capture d'écran") que vous ferez régulièrement le long du TP pour montrer la réussite ou l'échec de vos opérations. Rendez le tout dans un fichier Zip de même nom que précédemment. Pour tous les TP, ne validez votre dépôt sur Espadon qu'une fois le TP totalement terminé; à la fin de chaque séance, déposez un zip de vos travaux avec votre rapport en l'état, mettez-le à jour à chaque séance, et une fois le TP fini, validez.

I] Connexion

1) Connectez-vous à votre machine virtuelle Unix (Utilisez le mot de passe que vous avez défini au premier TP).

2) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS", créez un répertoire "Signaux" (commande *mkdir*; ce sera le répertoire de travail du TP).

II] Les signaux

1) Signaux.

Les signaux rapportent au programme l'occurrence d'un évènement exceptionnel. Parmi les raisons qui peuvent entrainer la *génération* ou encore la *levée* d'un signal, on trouve :

- Une erreur programme comme une division par zéro.
- Une requête utilisateur d'interrompre ou de de terminer un programme (comme avec l'emploi de Ctrl-z ou Ctrl-c.
- La terminaison d'un processus fils.
- L'expiration d'un Timer ou d'une Alarme.
- Un appel aux fonctions *kill* ou *raise* par le même processus.
- Un appel aux fonctions *kill* ou *raise* venant d'un autre processus.
- L'échec d'une tentative d'accès entrée/sortie.

Il y a les signaux synchrones (déclenchés par des actions programmes, comme une erreur de calcul), et asynchrones (déclenchées par des entités extérieures au programme). Tous les signaux levés passent en état suspendu jusqu'à ce qu'ils soient délivrés au programme concerné, qui peut alors soit les ignorer, soit spécifier la fonction qui doit alors être exécutée, soit accepter l'exécution de la fonction de gestion par défaut. La liste des signaux Unix peut être trouvée ici : http://en.wikipedia.org/wiki/Unix_signal.

2) Envoi.

Les signaux peuvent être envoyés dans un processus soit à partir de la fonction *raise(int SIG_TYPE)*, pour envoyer le signal au processus appelant, ou grâce à la fonction *kill(int pid, int SIG_TYPE)*, qui permet de spécifier le PID du processus destinataire.

3) Interception

Le mode d'interception le plus direct se fait avec la fonction *signal(int SIG_TYPE, void function)*, qui permet de déclarer la fonction *function* comme *callback* pour le signal spécifié. Si le signal spécifié est SIG_IGN, alors aucun callback ne sera exécuté à l'interception.

III] WTF ?

Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/Signaux", créez un répertoire "WTF".

2) Cherchez sur internet un exemple de programme sur les signaux Unix utilisant les fonctions ci-dessus, et les fichiers correspondants dans le répertoire *WTF* avec un fichier "makefile".

3) Corrigez le *makefile* pour compiler et exécuter cet exemple.

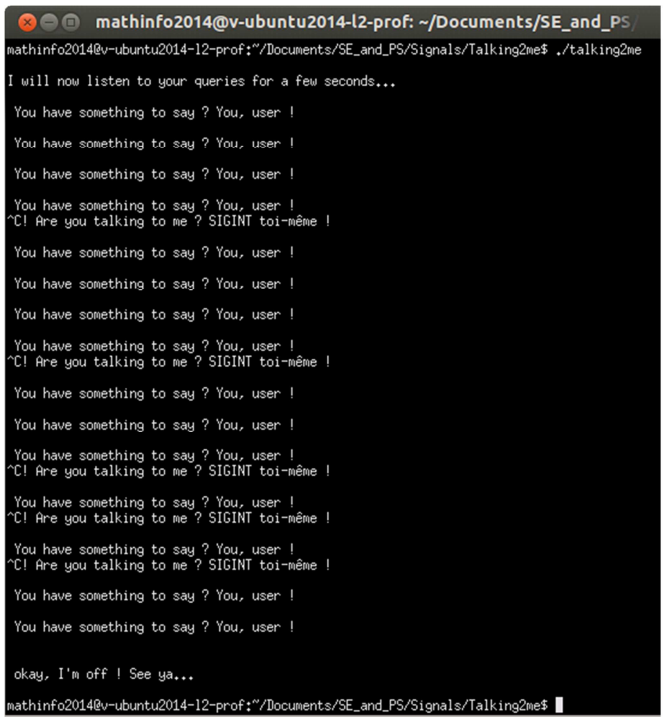
IV] Are you talking to me ?

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/Signaux", créez un répertoire "Talking2Me".

2) Créez un fichier " talking2Me.c" et un fichier "makefile".

3) Corrigez le makefile pour compiler Talking2Me.c.

4) Dans le fichier "Talking2Me.c", écrivez un programme en C qui intercepte le signal SIGINT (généré par Ctrl-c), et affiche un message toutes les secondes pendant quinze secondes invitant à effectuer la combinaison Ctrl-c au clavier. Le callback de SIGINT sera intitulé « void signalHandler(int signum) » et affichera un message pour acquitter son appel.



V] Catch me if you can

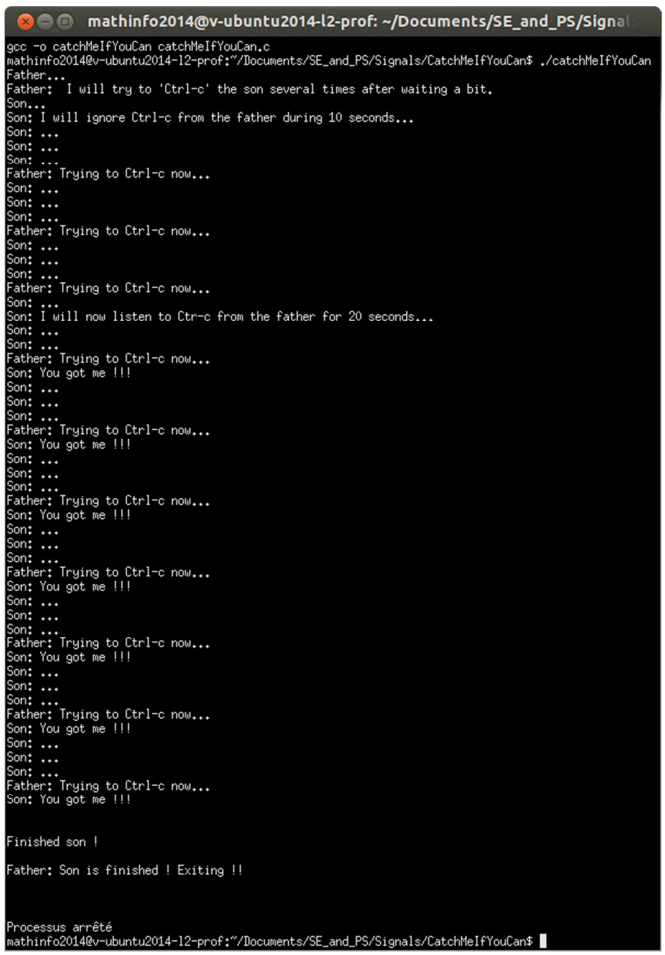
1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/Signaux", créez un répertoire "CatchMeIfYouCan".

2) Créez un fichier "catchMeIfYouCan.c" et un fichier "makefile".

3) Corrigez le makefile pour compiler catchMeIfYouCan.c.

4) Dans le fichier " catchMeIfYouCan.c", écrivez un programme en C qui se scinde en:

- un processus père, qui intercepte le signal SIGCHLD indiquant qu'un de ses fils est terminé, et affiche un message en tentant de générer des SIGINT toutes les 3 secondes à l'intention de son fils. Son callback 'fatherSigHandler' pour SIGCHLD acquitte son exécution par un message et met fin à l'exécution du processus père.



- un processus fils qui annule l'interception des SIGINT pendant 10 secondes, puis la rétablie et attends encore 20 secondes. Son callback 'sonSigHandler' affiche un message pour acquitter son exécution.

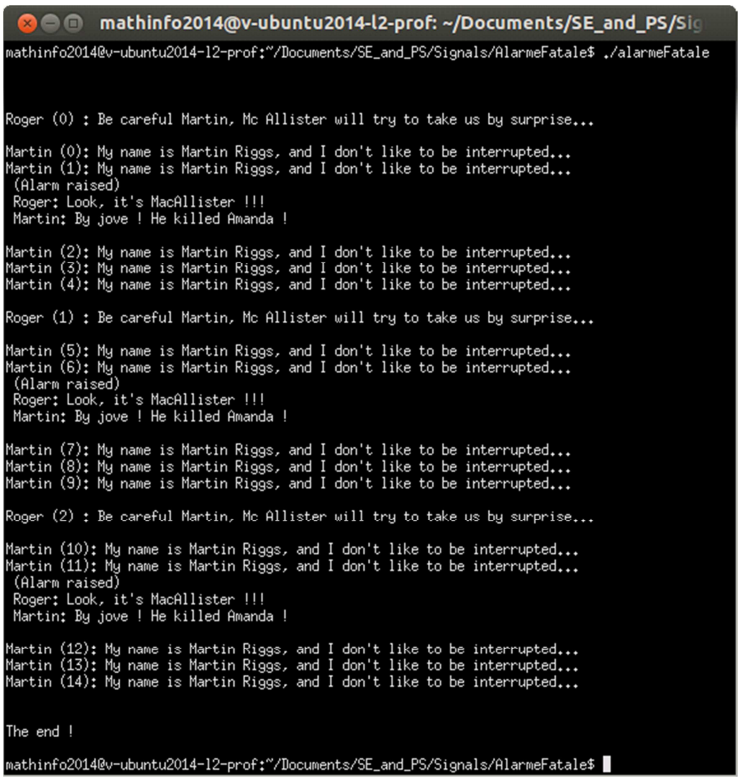
VI] Alarme fatale

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/Signaux", créez un répertoire "AlarmeFatale".

2) Créez un fichier "alarmeFatale.c" et un fichier "makefile".

3) Corrigez le makefile pour compiler alarmeFatale.c.

4) Dans le fichier "alarmeFatale.c", écrivez un programme en C qui intercepte le signal de l'alarme système (voir la fonction *alarm()*) SIGALRM. Le programme principal lancera une alarme prévue pour se déclencher dans les deux prochaines secondes, et commencera à Ecrire un texte toutes les secondes pendant 5 secondes vous pouvez répéter la boucle d'affichage plusieurs fois éventuellement). Le callback de l'alarme nommé 'alarmHandler' affichera un message quand il est appelé.



VII] Armagedon

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/Signaux", créez un répertoire "Armagedon".

2) Créez un fichier "armagedon.c" et un fichier "makefile".

3) Corrigez le makefile pour compiler armagedon.c.

4) Dans le fichier "armagedon.c", écrivez un programme en C qui se scinde en un programme père et des programmes fils. Les quatre fils interceptent le signal dit « utilisateur SIGUSR1 et affichent dans une boucle un message d'attente. Leur callback 'sonSigHandler' affiche un message puis provoque la levée du message SIGKILL pour lui-même. Le programme père affiche un message dans un boucle, et à la fin de la boucle déclenche le signal utilisateur pour tous ses fils et attends quelques secondes avant de terminer.