**Utilisation des appels systèmes en C/C++ - Gestion des signaux - Etat de la mémoire – synchronisation (6h)**

Gestion des signaux

|  |
| --- |
| * 1. Ouvrir un terminal et relever le PID du bash associé. Essayez de terminer le processus bash avec les signaux SIGTERM (n°15) et SIGINT (n°2). Conclure.
 |

Après avoir essayé de fermer le bash, on se rend compte que les signaux n’ont pas d’effet (le signal 2 semble avoir un léger effet au vu du saut de ligne)

|  |
| --- |
| * 1. Terminer le processus avec le signal SIGHUP (n°1)  Créer avec l'éditeur sublimeText les quatre fichiers suivants (langage C). La signification de leur contenu n'est à pas à analyser. Pour l’instant il suffit de savoir ce qu’ils font, sans avoir à comprendre comment ils le font.
 |

Tuer le bash avec le signal 1 marche

**sig1.c :**

#include <stdio.h>
int main()

1. {

  while (1)

 { printf ("sig1 en execution\n");

   sleep (2);

 }

 return 0;

}

1. *Après compilation et exécution, ce programme affiche toutes les 2 secondes le message « sig1 en execution » sur le terminal d’attachement. Il n’est pas protégé des signaux* ***SIGINT*** *et* ***SIGHUP***
2. **sig2.c :**
3. #include <signal.h>
4. #include <stdio.h>
5. int main()
6. {

 signal (SIGINT, SIG\_IGN);

 while (1)

 { printf ("sig2 en execution\n");

 sleep (2);

   }

 return 0;

1. }
2. *Ce programme est protégé du signal* ***SIGINT***

**sig3.c :**

#include <signal.h>

#include <stdio.h>

int main()

{

 signal (SIGINT, SIG\_IGN);

 signal (SIGHUP, SIG\_IGN);

 while (1)

 {

 printf ("sig3 en execution\n");

 sleep (2);

}

return 0;

}

*Ce programme est protégé des signaux* ***SIGINT*** *et* ***SIGHUP***

**sig4.c :**

#include <signal.h>

#include <stdio.h>

int main()

{

int leI;

for (leI = 1; leI <= NSIG; leI++)

 { signal (leI, SIG\_IGN);

}

 while (1)

{

 printf ("je suis en execution\n");

 sleep (2);

 }

return 0;

}

*Ce programme est (à priori) protégé de tous les signaux connus de LINUX ...*

|  |
| --- |
| * 1. Compiler ces fichiers avec la commande **gcc** ou **g++.**
 |

Commande : « gcc **nom\_du\_fichier\_source** -o **nom\_du\_fichier\_de\_sortie** »

|  |
| --- |
| * 1. Lancer le processus sig1 autant de fois que nécessaire pour répondre aux questions suivantes :
* Peut-on arrêter son exécution avec **CTRL+c** ?
* Peut-on arrêter son exécution avec la commande **$> kill *-SIGINT pid*** frappée depuis un autre terminal ? (pid est le numéro du processus **sig1**)
* Peut-on arrêter son exécution en fermant le terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
* Peut-on arrêter son exécution avec la commande **$> kill *-SIGHUP pid*** frappée depuis un autre terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
 |

* **CTRL+c** arrête l’exécution du programme
* **kill -SIGINT pid** arrête l’exécution du programme
* **Fermer le terminal** arrête l’exécution du programme
* **kill -SIGHUP pid** arrête l’exécution du programme, avec un message de sortie « Fin de la connexion (raccroché) »

|  |
| --- |
| * 1. Lancer le processus sig2 autant de fois que nécessaire pour répondre aux questions suivantes :
* Peut-on arrêter son exécution avec **CTRL+c** ?
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGINT* pid** frappée depuis un autre terminal ? (pid est le numéro du processus sig2)
* Peut-on arrêter son exécution en fermant le terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGHUP* pid** frappée depuis un autre terminal ? (Vérifier avec la commande ps -ef lancée depuis un autre terminal)
 |

* **CTRL+c** n’arrête pas l’exécution du programme
* **kill -SIGINT pid** n’arrête pas l’exécution du programme
* **Fermer le terminal** arrête l’exécution du programme
* **kill -SIGHUP pid** arrête l’exécution du programme, avec un message de sortie « Fin de la connexion (raccroché) »

|  |
| --- |
| * 1. Lancer le processus sig3 autant de fois que nécessaire pour répondre aux questions suivantes :
* Peut-on arrêter son exécution avec CTRL+c ?
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGINT* pid** frappée depuis un autre terminal ? (pid est le numéro du processus sig3)
* Peut-on arrêter son exécution en fermant le terminal ? (Vérifier avec la commande ps -ef lancée depuis un autre terminal)
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGHUP* pid** frappée depuis un autre terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGKILL*** pid frappée depuis un autre terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
 |

* **CTRL+c** n’arrête pas l’exécution du programme
* **kill -SIGINT pid** n’arrête pas l’exécution du programme
* **Fermer le terminal** n’arrête pas l’exécution du programme
* **kill -SIGHUP pid** n’arrête pas l’exécution du programme
* **kill -SIGKILL pid** arrête le processus

|  |
| --- |
| * 1. Lancer le processus sig4 autant de fois que nécessaire pour répondre aux questions suivantes :
* Peut-on arrêter son exécution avec CTRL+c ?
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGINT* pid** frappée depuis un autre terminal ? (pid est le numéro du processus sig4)
* Peut-on arrêter son exécution en fermant le terminal ? (Vérifier avec la commande **ps -ef** lancée depuis un autre terminal)
* Peut-on arrêter son exécution avec la commande **$> kill -*SIGHUP* pid** frappée depuis un autre terminal ? (Vérifier avec la commande ps -ef lancée depuis un autre terminal)
* Peut-on arrêter son exécu on avec la commande **$> kill -*SIGKILL* pid** frappée depuis un autre terminal ? (Vérifier avec la commande ps -ef lancée depuis un autre terminal)

Conclusion ?   |

* **CTRL+c** n’arrête pas l’exécution du programme
* **kill -SIGINT pid** n’arrête pas l’exécution du programme
* **Fermer le terminal** n’arrête pas l’exécution du programme
* **kill -SIGHUP pid** n’arrête pas l’exécution du programme
* **kill -SIGKILL pid** arrête le processus

Conclusion :

* **CTRL-C** et le signal **SIGINT** font la même chose
* **(\*signal\*, SIG\_IGN);** empêche le signal d’être attrapé
* **Fermer le terminal d’origine** reviendrait a envoyer le signal SIGHUP au programme
* **SIGKILL** tue tout les programmes, qu’importe les signaux bloqués

|  |
| --- |
| * 1. Lancer à nouveau le processus **sig1**
* Rester connecté en tant qu’utilisateur **invite** dans le premier terminal, et se connecter en tant qu’utilisateur **admini** via la commande **$> su admini** frappée depuis un **second terminal**.   $> kill -*SIGKILL* pid (pid est le numéro du processus **sig1**)   $> ps -ef
* Pourquoi le processus **sig1** n’a pas été détruit ?
 |

Les signaux ne doivent marcher que sur le même utilisateur