Debugging Système et Noyau
-
Travaux Pratique

Aurélien Cedeyn

2017-2018

1 Préparation

1. Vous disposez d'une machine virtuelle préparer pour suivre ce TP.
   — Utilisateurs :
      — root : debug
      — user : debuguser
2. À la fin de ce TP vous devrez rendre vos différentes réponses avec les sorties de vos commandes en suivant le format suivant :
   — Un répertoire à vos nom et prénoms
   — Un un fichier par question

Format des réponses

```
$ ls aurelien.cedeyn/ -l
total 0
-rw-rw-r-- 1 mat mat 0 nov.  27 22:47 2.1
-rw-rw-r-- 1 mat mat 0 nov.  27 22:47 3.1
-rw-rw-r-- 1 mat mat 0 nov.  27 22:47 3.2
-rw-rw-r-- 1 mat mat 0 nov.  27 22:47 3.3
-rw-rw-r-- 1 mat mat 0 nov.  27 22:47 3.4
```

Première partie
User space

2 L'espace utilisateur

1. Lister les processus de votre utilisateur
2. Afficher les fichiers lus par la commande ps
3. Afficher le nombre d'appels systèmes effectués par la commande *ps -elf*
4. Que fait la commande lsof ?
   — Utilisez un des outils vu précédemment en cours pour voir les appels systèmes effectués par cette commande.
   — Quels fichiers ouvre-t-elle ?
   — Quels sont les appels aux librairies externes effectués cette commande ?
   — Quels est la fonction la plus appelée par *lsof* ?
   — À quoi sert-elle ?

3 La pile

1. Écrire un programme en C qui dépasse la taille de la pile.
   — **Indice :** *ulimit -a* permet de connaître les restrictions du système
   — Quelles sont les différentes façons, selon vous, de dépasser la taille de la pile ?
   — Quelle erreur obtenez-vous ? Que signifie-t-elle ?
   — Comment corriger le programme ou l'environnement pour ne plus avoir cette erreur ?

4 La compilation/gdb

1. Compilez avec et sans les symboles de debug le programme C suivant
   Indice : Pour compiler avec les symboles de debug : gcc -g source.c -o binaire

infinite.c

```c
#include <stdlib.h>
#include <unistd.h>

int check(char cond){
    return(cond == 0);
}

void loop(void){
    int a=0;

    while(check(a == 0)){
        usleep(1000);
    }
}


int main(void) {
    loop();
    exit(0);
}
```

   — Quelles différences observez vous entre les deux binaires ?
   — Observez les symboles de débug avec la command *readelf*.
   Indice : *man readelf*

2. Lancer le programme compilé avec les symboles de debug via gdb
   — Affichez le code source dans gdb
   — Débuter son exécution
   — Interrompez-le et affichez sa pile d'appel

3. Prenez un corefile d'un processus sur la machine
   — Lancer gdb avec le corefile généré
   — Affichez la pile d'appel du processus.

4. Attachez-vous au processus crazy qui tourne sur la machine avec gdb
   — Affichez le code source dans gdb
   — Placez un point d'arrêt (breakpoint) sur à la ligne 12 de la fonction main.
   — Continuez le programme.

— Affichez la pile d'appel.
— Affichez la valeur de la variable *count*.
— Modifiez la pour que le programme se finisse.