Debugging User TP
-
Travaux Pratiques

Aurélien Cedeyn

2021 - 2022

Préparation

- Vous disposez d'une machine virtuelle *pcocc* pour suivre ce TP.
  - Template *pcocc* : debug
  - Configuration de la machine virtuelle : CentOS-7.5
  - Configuration *pcocc*

```
Lancez les commandes suivantes
[@hpc01 ~]$ /home/cedeyna/bin/mk_pcocc_debug
La configuration du template debug s'est correctement déroulée
[@hpc01 ~]$ pcocc alloc -n2 debug
salloc: Granted job allocation 13902
Configuring hosts... (done)
[@hpc01 ~]$ pcocc ssh root@vm0
```

- Ce TP est noté, il vous est demandé de rendre ce que vous avez pu réaliser à la fin de celui-ci.
- Vous avez jusqu'au vendredi 17/12 23h59 pour envoyer le compte rendu complet du TP.
- Ces deux échéances constitueront votre note de TP.
- Les différentes réponses avec les sorties de vos commandes devront être rendu sous forme de rapport au format :
  - Pour le 1e TP : nom.prenom.ASE_Debug_User.pdf ou nom.prenom.ASE_Debug_User.odt
  - Pour le 2e TP : nom.prenom.ASE_Debug_Kernel.pdf ou nom.prenom.ASE_Debug_Kernel.odt
  - Pour le 3e TP : nom.prenom.ASE_Debug_Optim.pdf ou nom.prenom.ASE_Debug_Optim.odt

User space

1 L'espace utilisateur

1. Listez les processus de votre utilisateur.
2. Affichez les fichiers lus par la commande *ps*.
3. Affichez le nombre d'appels systèmes effectués par la commande *ps -elf*
4. La commande *lsof -e /mnt*.
   - 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 par cette commande ?
   - Quels est la fonction la plus appelée par *lsof* ?
   - À quoi sert-elle ?

2 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 ?

3 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)){
        usleep(1000);
    }
}

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

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

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

3. Choisissez un processus sur la machine et prenez en un corefile.

— Lancez gdb avec le corefile généré.
— Affichez la pile d'appel du processus.
— Quittez *gdb*.

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.