Міністерство освіти України
Харківський національний університет радіоелектроніки

Звіт до лабораторної роботи № 4
з курсу «Операційні системи UNIX»

Виконав: ст. гр. ПЗПІ-16-3
Губар С.О.
Прийняв: Сокорчук І.П.

Харків 2020

Мета роботи: познайомитися з процесами, сигналами у операційних система UNIX. Написати скрипт, який запускає ще один процес, та взаємодіє з ним за допомогою сигналів.

Хід виконання роботи

Створимо файл lab4.sh, зробимо його виконуваним. Додамо код запуску дочірнього процесу за допомогою круглих дужок з опцією & для того, щоб він виконувався у фоні та не завершував виконання при завершенні батьківського процесу:

```
while true; do
   echo "Child is working..." 1>/dev/null
   sleep 5
done
) &
```

Запам’ятаємо ідентифікатори процесів:

```
child=$!
current_pid=$$
```

Використаємо команду trap для того, щоб задати функції, які будуть обробляти сигнали SIGTERM, SIGINT та SIGUSR2 у батьківському процесі:

```
trap parent_term SIGTERM
trap parent_interruption SIGINT
trap parent_ping SIGUSR2
```

Напишемо код обробки сигналів

```
kill_child_process() {
   kill -SIGTERM "$child"
   echo "Killed child process"
}
```

```
parent_term() {
   kill -SIGTERM "$child"
      log_message "${current_pid}; 15; SIGTERM; parent exits and throws SIGTERM to subshell"
   echo "Sent sigterm to child ${child}"
   exit 1
}

parent_interruption() {
   log_message "${current_pid}; 2; SIGINT; parent got interrupted"
   exit 1
}

parent_ping() {
     log_message "${current_pid}; 12; SIGUSR2; nothing, parent confirms that child process pinged back"
}
```

Реалізуємо функцію логування

```
log_message() {
   current_date=$(date)
   timestamp=$(date +%s)
   message=$(printf "%s; %s; %s" "$current_date" "$timestamp" "$1")
   logger $message
}
```

Додамо аналогічні функції у дочірній процес:

```
(
child_term() {
   pid=$(exec sh -c 'echo "$PPID"')
   log_message "${pid}; 15; SIGTERM; child process terminated"
   exit 1
}
```

```
custom_signal() {
    pid=$(exec sh -c 'echo "$PPID"')
    log_message "${pid}; 10; SIGUSR1; just ping signal"
}

ping_back_signal() {
    pid=$(exec sh -c 'echo "$PPID"')
    log_message "${pid}; 12; SIGUSR2; ping father back"
    kill -SIGUSR2 $$
}

heartbeat() {
    pid=$(exec sh -c 'echo "$PPID"')
    log_message "${pid}; 18; SIGCONTv; heartbeat"
}
```

Та обробку сигналів:

```
trap child_term SIGTERM
trap custom_signal SIGUSR1
trap ping_back_signal SIGUSR2
trap heartbeat 18
```

Реалізуємо інтерактивну взаємодію з користувачем за допомогою функції read:

```
while read -r -p "What you wanna do? For options - open --help " && [[ $REPLY != q ]]; do
  case $REPLY in
    test) send_test_signal_to_child;;
    exit) kill_child_process;;
    back) send_ping_back_signal_to_child;;
    *) echo "Try Again.";;
  esac
done
```

Та додамо автоматичне посилання сигналу до дочірнього процесу за допомогою ще одного процесу:

```
(
   while true; do
      kill -18 "$child" 2>/dev/null
      sleep 3
   done
) &
```

Реалізуємо функцію help:

```
if [ "$1" = "--help" ] || [ "$1" = "-h" ]; then
   echo "Usage ./pzpi-16-3-hubar-serhii-lab4.sh [-h | --help]";
   echo "Script launches child process and waits for interactive communication with user";
   echo "Possible interaction options:";
   echo "1) test - send ping signal to child subshell";
   echo "2) kill - to kill child subshell"
   echo "3) back - to send ping back (child subshell will respond)"
   echo "q to quit"
   exit 0;
fi
```

І перевіримо, що у скрипт не передаються аргументи, за винятком --help:

```
if [ "$#" -ne 0 ]; then
   if [[ $LANG =~ uk_UA ]]; then
      echo "Скрипт не може приймати аргументів крім --help" 1>&2
   else
      echo "Script can't be run with arguments (excuding --help)" 1>&2
   fi
   exit 1
fi
```

Висновки: у ході виконання лабораторної работи було проведено ознайомлення з процесами, та взаємодією між процесами за допомогою сигналів.