ใบงาน 8a

วัตถุประสงค์ เพื่อศึกษาวิธีการรับ/ส่ง signal ระหว่างโพรเซสบนระบบ unix และ วิธีการรับ/ส่ง ข้อมูลระหว่างโพรเซสผ่าน unix pipe

คำสั่ง

Q1 (2.4)

Q2 (3.3)

กิจกรรม

1. ศึกษา flow มาตรฐานการทำงานของ signal ผ่าน interrupt handler

1.1 override default operation ให้ handler ที่เราสร้างทำงานแทน (line 10)

1.2 disable การรับ signal นั้นๆ (line 18) จนกว่า handler เราทำงานเสร็จ

1.3 ติดตั้ง handler ของเราให้ทำงานต่อ (line 28) หากต้องการ

1.4 ตัวอย่างนี้ ใช้ Ctrl-C ซึ่งส่ง SIGINT ไปยังระบบปฏิบัติการ (ซึ่งปกติ จะ break การทำงาน)

```
/*  This is a simple program that
   illustrates the use of signal handler.
   It catches Ctrl-C (Ctrl-C generates SIGINT).
  */
1  #include <stdio.h>
2  #include <unistd.h>    // for pause
3  #include <signal.h>
4  #include <stdlib.h>    // for exit
5
6  void  INThandler(int);  // prototype
7
8  void  main(void) {
9        /* install Ctrl-C handler  */
10       signal(SIGINT, INThandler);
11
12       while (1) /* loop forever while pause */
13           /* cpu can be allocated to others */
14           pause();
15 } //main
16 void  INThandler(int sig) {
17   /* when an SIGINT is generated,
18       Start here as installed (declared)  */
19   signal(sig,  SIG_IGN);   /* disable Ctrl-C */
20   /* response from INThandler */
21   printf("Did you hit Ctrl-C?\n
22            want to quit? [y/n] \n");
23   char c;
24   //c = getchar(); /* requires to trim \n */
25   scanf("%c",&c);
26   if (c == 'y' || c == 'Y') /* if y or Y */
27     exit(0);
28   else
29     signal(SIGINT, INThandler);
30     /* reinstall the handler */
31 }
```

2. raise()

2.1 raise() (line 27) ส่ง signal ให้ เพื่อให้ handler ทำงาน

2.2 ตัวอย่างนี้ใช้ SIGUSR1

2.3 ตัวอย่างนี้ไม่จำเป็นต้องกลับไป install handler

2.4 Q1. เขียน Output ของ line 13 (จะมี compiler ของเครื่องไหนได้ขนาดของ long แปลกกว่าเครื่องอื่น?)

(เขียน Q1 ไว้เป็น comment ใต้รหัสนักศึกษา)

```
1  /*  This program demonstrates the use of ANSI C
2  library function raise(). Detecting overflow */
3  #include  <stdio.h>
4  #include  <stdlib.h>
5  #include  <signal.h>
6
7  signed long  prev_computed, i;
8  /* global variables */
9
10 void  SIGhandler(int sig) { // no need prototype
11   printf("\nReceived a ");
12   printf("SIGUSR1.  The max n is ");
13   printf("%ld! = %ld\n", i-1, prev_computed);
14   exit(0);
15 }
16 int  main(void) {
17   signed long  cur_value;
18
19 //to know type size, printf("%ld\n",sizeof(long));
20   printf("2 power n:\n");
21   signal(SIGUSR1, SIGhandler);
22   /* install SIGUSR1 handler */
23   prev_computed = 1;
24   for (i = 1;   ; i++ ) { // loop until raise()
25     cur_value = prev_computed * i;// 2 to the n
26     if (cur_value < prev_computed)
27       raise(SIGUSR1);    /* overflow, raise */
28     prev_computed = cur_value;
29    /* after raise prev is the last value possible */
30   } //for
31   Return 0;
32 }  //main
```

3. kill()

3.1 kill() system call (line 22) ส่ง signal ให้ โพรเซสที่ระบุ โดย โพรเซสที่ได้รับจะ เป็นผู้ handle เช่นโปรแกรมตัวอย่าง (เปลี่ยน notDone จาก 1 เป็น 0

3.2 unix ใช้คำสั่ง kill -9 pid เพื่อให้โปรเซส (pid) จบการทำงาน ความหมายจริงๆ คือ kill -SIGKILL pid โดยใช้คำสั่ง kill เป็นตัวส่ง signal เหตุที่เรียกคำสั่งนี้ว่า kill เพราะ default signal คือ SIGKILL (ค่าของ SIGKILL คือ 9)

3.3 Q2 เขียนโปรแกรมให้ลูกวนลูปจนกว่าแม่ส่ง SIGKILL มาให้ลูกเลิกทำงาน (แสดง printf() ไว้ใต้ลูปเพื่อแสดงว่าลูกไปไม่ถึง

- ไม่ต้อง override (ดังนั้นไม่ต้อง install) เพราะเราต้องการ SIGKILL default handler

```c
/* Program demonstrates the use of kill() */
#include  <stdio.h>
#include  <stdlib.h>
#include  <signal.h>
#include  <stdlib.h>

int notDone = 1; int cnt = 0;
void  mySIGhandler(int sig) {
  signal(SIGALRM,SIG_IGN);
  //breaks infinite loop by reset notDone
  notDone = 0;
  //printf("loop should break\n");
}
int  main(void) {
   //install handler
   signal(SIGALRM,mySIGhandler);
   pid_t pid = fork();
   if (pid == 0) {
  //let parent enters loop (manual synchronization)
     sleep(4);   //let parent enters loop
     printf("sending SIGALRM\n");
     kill(getppid(),SIGALRM);
     exit(0);  // send signal and die
   } else {
     printf("parent wait for SIGALRM\n");
     while (notDone) cnt++; //infinite loop
   }
   //Could be time instead of cnt
   printf("it takes %d\n",cnt);
   return 0;
} //main
```

```
suntana@DESKTOP-IQOCR48:~/lab8a$ ./q3
parent wait for SIGALRM
sending SIGALRM
it takes 1783640651
```

```c
    pid_t pid = fork();
    if (pid == 0) {
        printf("child created\n");
        while (1);              //infinite loop
        printf("this line should not be shown\n");
        exit(0);
    } else {
```

กำหนดส่ง (TBA)