SCI19 2243 Operating Systems (1/2568)

สาขาวิชาคณิตศาสตร์และภูมิสารสนเทศ สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

รหัสนักศึกษา B6638146 ชื่อ-สกุล เจษฎา ธัญญะ

Assignment ครั้งที่ 6 – Signal and Pipe

วัตถุประสงค์ของการปฏิบัติการ

- สามารถส่งและจัดการกับ Signal ที่เกิดกับ Process ได้ (IPC Signal)
- สามารถ รับ/ส่ง ข้อมูลระหว่าง Process ผ่าน UNIX Pipe ได้ (IPC Message Passing)

1. ศึกษาการทำงาน signal(), raise() และ kill()

number	hex	symbol	description
1	0x01	SIGHUP	Hangup
2	0x02	SIGINT	Interrupt
3	0x03	SIGQUIT	Quit
4	0x04	SIGILL	Illegal instruction
5	0x05	SIGTRAP	Trace/breakpoint trap
6	0x06	SIGABRT	Aborted
6	0x06	SIGIOT	*(Same value as SIGABRT)* Aborted
7	0x07	SIGBUS	Bus error
8	0x08	SIGFPE	Floating point exception
9	0x09	SIGKILL	Killed
10	0x0a	SIGUSR1	User defined signal 1
11	0x0b	SIGSEGV	Segmentation fault
12	0x0c	SIGUSR2	User defined signal 2
13	0x0d	SIGPIPE	Broken pipe
14	0x0e	SIGALRM	Alarm clock
15	0x0f	SIGTERM	Terminated

รูปที่ 1 ตารางแสดง Unix/Linux Signal 1 – 15

(สามารถดู Signal อื่น ๆ ได้เพิ่มเติมที่ https://chromium.googlesource.com/chromiumos/docs/+/master/constants/signals.md)

อาจารย์: อนุพงษ์ บรรจงการ

29/08/2568

แก้ไข Code ที่ 1 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.1 - 1.2

Code ที่ 1 (Signal: Catch and use OS default)

No.	File Name: osLab6_sig01.c
1	#include <signal.h>
2	void main() {
3	signal(SIGINT, SIGhandler); // Register Ctrl-C handler
4	while (1) {
5	printf("Going to sleep for a second ...\n");
6	sleep(1);
7	}
8	}
9	void SIGhandler(int sig) {
10	printf("Caught signal %d, coming out ...\n", sig);
11	exit(0);
12	}

1.1. เมื่อรันโปรแกรม - แล้วต้องการจะออกจากโปรแกรมตามที่กำหนดไว้จะต้องกด Short Key บน Keyboard อย่างไร และได้ค่า Signal หมายเลขใด..ต้องกด Ctrl-C บนคีย์บอร์ด ซึ่งจะส่ง Signal หมายเลข 2 (SIGINT).....

1.2. หากต้องการให้ handler function ตรวจจับการออกจากโปรแกรมด้วย Short Key “Ctrl-\” จะต้องเปลี่ยน Code ในบรรทัดที่ 3 อย่างไร..signal(SIGQUIT, SIGhandler);.....

แก้ไข Code ที่ 2 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.3 - 1.8

Code ที่ 2 (Signal: Catch)

No.	File Name: osLab6_sig02.c
1	void SIGhandler(int sig) {
2	switch (sig) {
3	case 2:
4	printf("Ha Ha Ha ... you can not interrupt me ;P\n");
5	break;
6	case (.....1.3.....):
7	printf("Ha Ha Ha ... you can not quit me ;P\n");
8	}
9	}
10	int main(void) {
11	/* Register Quit handler */

```
12          if (signal((.....1.4.....), SIGhandler) == SIG_ERR) {
13                  printf("\ncan't catch SIGQUIT\n");
14          }
15          /* Register Interrupt handler */
16          if (signal(SIGINT, (.....1.5.....)) == SIG_ERR) {
17                  printf("\ncan't catch SIGINT\n");
18          }
19          while(1) {
20                  sleep(1);
21          }
22          return(0);
23  }
```

1.3. SIGQUI T .....

1.4. SIGQUI T .....

1.5. SIGhandler .....

1.6. รันโปรแกรม - หากต้องการออกจากโปรแกรมด้วย Short Key “Ctrl-C” ได้ผลลัพธ์อย่างไร (แสดงผลลัพธ์)

```
^CHa Ha Ha ... you can not interrupt me ;P
```

1.7. รันโปรแกรม - หากต้องการออกจากโปรแกรมด้วย Short Key “Ctrl-\” ได้ผลลัพธ์อย่างไร (แสดงผลลัพธ์)

```
^\Ha Ha Ha ... you can not quit me ;P
```

1.8. รันโปรแกรม – จะออกจากโปรแกรมได้อย่างไรโดยที่ **ห้ามกดปิดโปรแกรม Terminal** อธิบายขั้นตอน .....

เปิด Terminal ใช้คำสั่ง kill -9 <PID> แล้วกด Enter .....

```
jeadsada@Salmon:/mnt/c/Users/fewku$ kill -9 518
Killed
jeadsada@Salmon:/mnt/c/Users/fewku$
```

.....

.....

แก้ไข Code ที่ 3 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.9 - 1.12

Code ที่ 3 (Signal: Mask)

No.	File Name: osLab6_sig03.c
1	int main(void) {
2	signal((.....1.9.....)); // Register Interrupt Handler
3	while(1);
4	pause();

```
5       return 0;
6   }
7   void INThandler(int sig) {
8       signal((.....1.10.....), SIG_IGN);          // Mask Signal
9       printf("\nCtrl-C was hit !!!\nWant to quit? [y/n]\n");
10      char input;
11      input = getchar();
12      if (input == 'y' || input == 'Y') {
13          exit(0);
14      } else {
15          (.....1.11.....)  // Reregister Interrupt Handler
16      }
17  }
```

1.9. SIGINT, INThandler

1.10. SIGINT, SIG_IGN

1.11. SIGINT, INThandler

1.12. บรรทัดที่ 11 สามารถใช้วิธีอื่นในการรับ Input จาก Keyboard ได้หรือไม่ อย่างไร ได้

scanf(): ใช้ในการรับข้อมูลตามรูปแบบที่กำหนด สามารถใช้รับตัวอักษร 1 ตัวได้เช่นกัน

fgets(): รับข้อความทั้งบรรทัดเข้ามาเก็บในตัวแปรแบบอาร์เรย์ (string) แล้วค่อยดึงตัวอักษรตัวแรกมาใช้

แก้ไข Code ที่ 4 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.13 - 1.16

Code ที่ 4 (raise() function)

No.	File Name: osLab6_sig04.c
1	signed long prev, i; // Global Variables
2	void (.....1.13.....) (int sig) {
3	printf("Received SIGUSR1. The max is %ld! = %ld\n", i-1, prev);
4	exit(0);
5	}
6	int main(void) {
7	signed long curr;
8	(.....1.14.....) // Register SIGUSR1 Handler
9	for (prev = i = 1; ; i++, prev = curr) {
10	curr = prev * i;
11	if (curr < prev) {

```
12                      raise(.....1.15.....)); // Raise SIGUSR1 to Handler
13              } else {
14                      printf(" %ld! = %ld (%ld)\n", i, curr, prev);
15              }
16      }
17      return 0;
18  }
```

1.13. USR1handler .....

1.14. signal(SIGUSR1, USR1handler); .....

1.15. SIGUSR1 .....

1.16. รันโปรแกรมและแสดงผลลัพธ์ของบรรทัดที่ 3 

.....

.....

แก้ไข Code ที่ 5 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.17 - 1.23

Code ที่ 5 (kill() function - client send signal to parent)

No.	File Name: osLab6_sig05.c
1	#define SLEEP 3
2	int flag = 1; int count = 0; // Global Variables
3	void mySIGhandler(int sig) {
4	signal(SIGUSR2, (.....1.17.....)); // Mask Signal
5	flag = 0;
6	}
7	void main() {
8	clock_t start; pid_t pid;
9	signal(SIGUSR2, mySIGhandler); // Register Signal Handler
10	if ((pid = fork()) < 0) { // Fork process
11	printf("fork failed"); exit(1);
12	} else if ((.....1.18.....)) { // Child process
13	sleep(SLEEP);
14	printf("Child sending signal ...\n");
15	kill((.....1.19.....), SIGUSR2); // Send signal to parent process
16	exit(0);
17	} else { // Parent process

```
18          printf("Parent waiting signal ...\n");
19          start = clock();
20          while ((.....1.20.....)) count+=1;
21          printf("It takes %d loops\n", count);
22          printf("It takes %.2f sec.\n", (double)((clock() - start) / CLOCKS_PER_SEC));
23      }
24  }
```

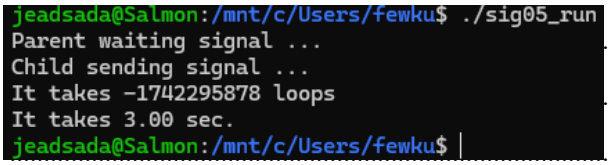
1.17. SIG_IGN

1.18. pid == 0

1.19. getppid()

1.20. flag

1.21. รันโปรแกรมและแสดงผลลัพธ์จากการรัน



1.22. หากเปลี่ยน Signal จาก “SIGUSR2” เป็น “SIGINT” โปรแกรมจะให้ผลลัพธ์จากการรันเหมือนเดิมหรือไม่ เพราะอะไร เหมือนเดิม SIGNT (Interrupt Signal) เป็นสัญญาณที่สามารถดักจับ (catch) และกำหนดตัวจัดการสัญญาณ (signal handler) ได้

1.23. หากเปลี่ยน Signal จาก “SIGUSR2” เป็น “SIGKILL” โปรแกรมจะให้ผลลัพธ์จากการรันเหมือนเดิมหรือไม่ เพราะอะไร ไม่เหมือนเดิม mySIGhandler จะไม่ถูกเรียกใช้งาน และคำสั่ง printf ที่อยู่ท้ายโปรแกรมเพื่อแสดงผลลัพธ์การวนลูปและเวลาจะไม่ถูกประมวลผลเลย

แก้ไข Code ที่ 6 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 1.24 - 1.25

Code ที่ 6 (kill() function - parent send signal to client)

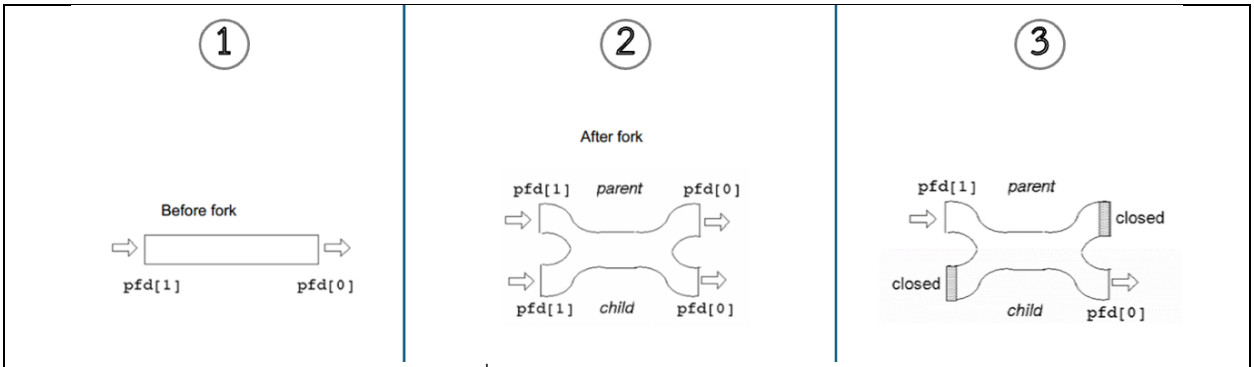
No.	File Name: osLab6_sig06.c
1	int c_pid; // Global Variable
2	void main() {
3	pid_t pid;
4	if ((pid = fork()) < 0) { // Fork process
5	printf("fork failed");
6	exit(1);
7	} else if (pid == 0) { // Child process
8	c_pid = (.....1.24.....);
9	printf("Child created\n");
10	while (1);
11	printf("Hello from child :)\n");
12	} else { // Parent process
13	printf("Parent created\n");
14	sleep(3);
15	printf("Hello from parent :)\n");
16	kill(c_pid, SIGKILL); // Send kill signal to child process
17	}
18	}

1.24. getpid() .....

1.25. จาก Code ที่ 6 Child process จัดการ Signal แบบใด (Default, Catch, Mask) Default .....

.....

2. ศึกษาการทำงานของ Unix Pipe (IPC Massage Passing)



รูปที่ 2 แสดงการทำงานของ Pipe

แก้ไข Code ที่ 7 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 2.1 - 2.7

Code ที่ 7 (pipe() - One way direction)

No.	File Name: osLab6_pip01.c
1	#define SIZE 10
2	void main() {
3	int nread, pfd[2];
4	char read_buf[SIZE], write_buf[SIZE * 100];
5	if ((.....2.1.....) == -1) { // Create pipe
6	perror("pipe failed\n");
7	exit(-1);
8	}
9	if (fork() == 0) { // Child Process
10	close((.....2.2.....)); // Close to write (output)
11	int cnt = 0;
12	while ((nread = read(pfd[0], read_buf, SIZE)) != 0) {
13	read_buf[READ] = '\0';
14	printf("Child read at round %d: %s\n", cnt+=1, read_buf);
15	}
16	(.....2.3.....)
17	} else { // Parent Process
18	(.....2.4.....) // Close to read (input)
19	strcpy(write_buf, "Hello ... CS ... SUT ... ARE YOU HAPPY?");
20	write(pfd[1], write_buf, strlen(write_buf) + 1);
21	close((.....2.5.....));
22	printf("Parent write done !!!\n");
23	wait(0); // Wait to sync return prompt when exit program
24	}
25	}

2.1. pipe(pfd)

2.2. pfd[1]

2.3. close(pfd[0]);

2.4. close(pfd[0]);

2.5. pfd[1]

2.6. Code บรรทัดที่ 13 มีหน้าที่อะไร จงอธิบาย (ทดสอบ Comment และ Uncomment รันโปรแกรมและศึกษาผลลัพธ์) .....

มีหน้าทีแปลงข้อมูลทีอ่านได้จาก pipe ให้กลายเป็น string ทีสมบูรณ์ในภาษา C โดยการเติมตัวอักษร null (\0) ต่อท้ายข้อมูล

2.7. แสดงผลลัพธ์จากการรันโปรแกรม .....

```
jeadsada@Salmon:/mnt/c/Users/fewku$ ./pip01_run
Parent write done !!!
Child read at round 1: Hello ...
Child read at round 2: CS ... SUT
Child read at round 3:  ... ARE Y
Child read at round 4: OU HAPPY?
```

```
jeadsada@Salmon:/mnt/c/Users/fewku$ ./pip01_run
Parent write done !!!
Child read at round 1: Hello ... Gr
Child read at round 2: CS ... SUTGr
Child read at round 3:  ... ARE YGr
Child read at round 4: OU HAPPY?
```

แก้ไข Code ที่ 8 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 2.8 - 2.12

Code ที่ 8 (pipe() - Two way direction)

No.	File Name: osLab6_pip02.c
1	#define SIZE 1024
2	void main() {
3	int pid, nread, pfd1[2], (.....2.8.....); char buf[SIZE];
4	if (pipe(pfd1) == -1) { // Create pipe#1
5	perror("pipe failed"); exit(1);
6	}
7	if (pipe(pfd2) == -1) { // Create pipe#2
8	perror("pipe failed"); exit(1);
9	}
10	if ((.....2.9.....) < 0) { // Fork process
11	perror("fork failed"); exit(2);
12	}
13	if (pid == 0) { // child process
14	/* Close pipe#1 to write and pipe#2 to read */
15	(.....2.10.....)
16	/* Pipe Receive */

```
17              while ((nread = read(pfd1[0], buf, SIZE)) != 0) {
18                      printf("Child read:\t %s\n", buf);
19              }
20              close(pfd1[0]);
21              /* Pipe Send */
22              strcpy(buf, "I am fine thank you :)");
23              write(pfd2[1], buf, strlen(buf) + 1);
24              close(pfd2[1]);
25      } else { // parent process
26              /* Close pipe#1 to read and pipe#2 to write */
27              close(pfd1[0]); close(pfd2[1]);
28              /* Pipe Send */
29              strcpy(buf, "How are you ?");
30              write(pfd1[1], buf, strlen(buf) + 1);
31              close(pfd1[1]);
32              /* Pipe Receive */
33              while ((nread = read((.....2.11.....)) != 0) {
34                      printf("Parent read:\t %s\n", buf);
35              }
36              close(pfd2[0]);
37      }
38  }
```

2.8. pfd2[2] .....

2.9. pid = fork() .....

2.10. close(pfd1[1]); close(pfd2[0]); .....

2.11. pfd2[0], buf, SIZE .....

2.12. แสดงผลลัพธ์จากการรันโปรแกรม .....

```
jeadsada@Salmon:/mnt/c/Users/fewku$ ./pip02_run
Child read:      How are you ?
Parent read:     I am fine thank you :)
```

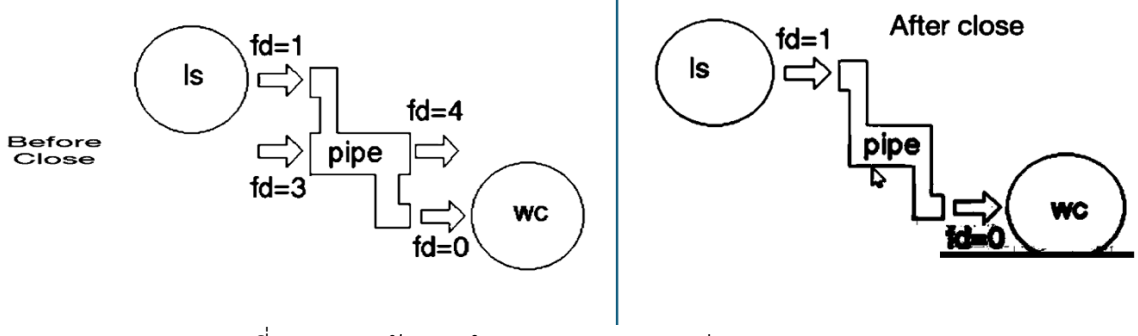
.....

.....

.....

.....

.....



รูปที่ 3 แสดงหลักการทำ duplicate pipe ผ่าน dup2() function

แก้ไข Code ที่ 9 ให้สามารถ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์และตอบคำถาม 2.12 - 2.19

Code ที่ 9 (pipe() with dup2())

No.	File Name: osLab6_pip03.c
1	void main() {
2	int pid, pfd[2];
3	if ((.....2.12.....)) { // Create pipe
4	perror("pipe failed"); exit(1);
5	}
6	if ((.....2.13.....) < 0) { // Fork process
7	perror("fork failed"); exit(2);
8	} else if (pid == 0) { // Child process
9	(.....2.14.....) // Close pipe input
10	dup2(pfd[1], 1); // Duplicate pipe output for exec()
11	close(pfd[1]); // Close pipe output
12	execlp("ls", "ls", NULL); // Exec ls program
13	perror("ls failed"); exit(3);
14	} else { // Parent process
15	close(pfd[1]); // Close pipe output
16	(.....2.15.....) // Duplicate pipe input for exec()
17	close(pfd[0]); // Close pip input
18	execlp(.....2.16.....)); // Exec wc program
19	perror("wc failed"); exit(4);
20	}
21	exit(0); // End program
22	/* Parent process wait child process sync by OS */
23	}

2.13. pipe(pfd) == -1

2.14. pid = fork()

2.15. close(pfd[0]);

2.16. dup2(pfd[0], 0);

2.17. "wc", "wc", NULL

2.18. ผลลัพธ์จากการรันโปรแกรม เปรียบเสมือนกับการรัน Unix/Linux Command แบบใด ls | wc

2.19. แสดงผลลัพธ์จากการรันโปรแกรมพร้อมอธิบายความหมายของผลลัพธ์

```
jeadsada@Salmon:/mnt/c/Users/fewku$ ./pip03_run
     86      91    1159
```

จำนวนบรรทัด จำนวนคำ จำนวนไบต์

2.20. พัฒนาโปรแกรมภาษา C โดยใช้ System Call fork() และ exec() ที่ Process ใช้ IPC แบบ Message Passing ในการส่งข้อมูลระหว่างกัน

- โปรแกรมสามารถทำงานได้ผลลัพธ์เหมือนกับการรัน Unix/Linux Command ดังต่อไปนี้

  “cat /etc/passwd | grep ^root”

- แสดง Source Code ของโปรแกรมภาษา C ที่ได้พัฒนาลงในช่องว่างด้านล่าง

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    int     pfd[2];
    pid_t   pid;

    if (pipe(pfd) == -1) {
        perror("pipe failed");
        exit(EXIT_FAILURE);
    }

    pid = fork();
    if (pid < 0) {
        perror("fork failed");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        close(pfd[0]);
        dup2(pfd[1], STDOUT_FILENO);
        close(pfd[1]);
        execlp("cat", "cat", "/etc/passwd", NULL);
        perror("execlp 'cat' failed");
        exit(EXIT_FAILURE);
    }
    else {
        close(pfd[1]);
        dup2(pfd[0], STDIN_FILENO);
        close(pfd[0]);
        execlp("grep", "grep", "^root", NULL);
        perror("execlp 'grep' failed");
        exit(EXIT_FAILURE);
    }

    wait(NULL);

    return 0;
}
```

```
jeadsada@Salmon:/mnt/c/Users/fewku$ ./cat_grep
root:x:0:0:root:/root:/bin/bash
jeadsada@Salmon:/mnt/c/Users/fewku$ cat /etc/passwd | grep ^root
root:x:0:0:root:/root:/bin/bash
```