05506007 Operating Systems (1/2568)

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

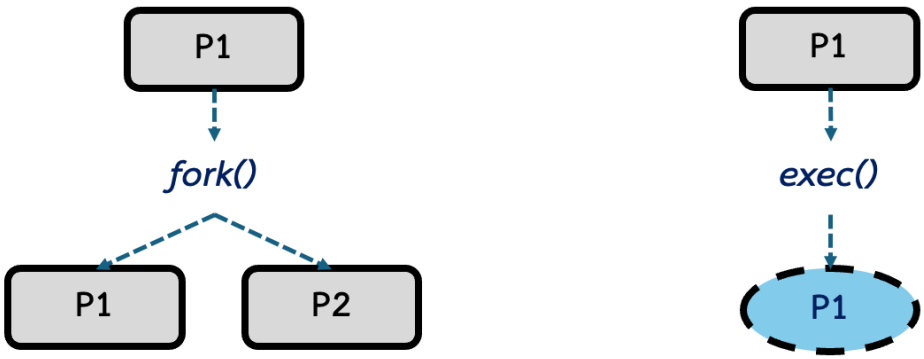
รหัสนักศึกษา 67050200 ชื่อ-สกุล **ธนกร บุญสินธุ์**

ปฏิบัติการ ครั้งที่ 5 – System Call fork() and exec()

วัตถุประสงค์ของการปฏิบัติการ

- อธิบายการทำงานของ System Call fork() และ exec() ได้
- ใช้งาน System Call fork() และ exit() ได้

1. System Call: fork(), wait(), และ exit()



รูปที่ 1 แสดงการสร้าง Process ของ System call fork() และ exec()

System Call “fork()” เป็นการสร้าง Process ใหม่โดยการ Copy Process โดย Process ที่เรียก fork() จะเป็น Parent Process และ Process ที่ถูกสร้างจะเป็น Child Process (ทั้งสองเป็น Process ที่ทำงานเหมือนกัน) ส่วน System Call ตระกูล “exec()” จะเป็นการสร้าง Process ใหม่โดยการ Replace Process เดิม กล่าวคือ เมื่อมีการเรียก System Call ตระกูล exec() Process ผู้เรียกจะเปลี่ยนการทำงานไปเป็นการทำงานแบบใหม่ตามที่เรียก แสดงการสร้าง Process ของ System Call ทั้งสองแบบ ดังรูปที่ 1

ทดสอบรัน Code ที่ 1 - แก้ไข Code ให้ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์ และตอบคำถาม 1.1 - 1.4

Code ที่ 1

No.	File Name: osLab5-01.c
1	void main () {
2	printf("#1 P_PID: %d", getpid());
3	pid_t id = fork();
4	if (id == 0) {
5	printf("#2 C_PID: %d", getpid());
6	printf("#3 P_PID: %d", getppid());
7	sleep(1);
8	exit(0);
9	} else {
10	printf("#4 P_PID: %d", getpid());
11	printf("#5 C_PID: %d", id);
12	sleep(1);
13	}
14	printf("HELLO CS KMITL");
15	}
หมายเหตุ: getppid() ใช้สำหรับเรียกค่า Process ID ของ Parent Process	

1.1. จงแสดงผลลัพธ์จากการรันโปรแกรม .....

#1 P_PID: 358846#4 P_PID: 358846#5 C_PID: 358847HELLO CS KMITL#1 P_PID: 358846#2 C_PID: 358847#3 P_PID: 358846

.....

.....

.....

.....

1.2. บรรทัดที่ 7 และ 12 เปลี่ยนค่า Parameter ของ sleep() Function จาก 1 เป็น 0 ทำการ Compile Code และทดสอบรัน สังเกตผลลัพธ์ที่เปลี่ยนแปลง พร้อมอธิบายเหตุผล .....

code ทำงานได้ไวขึ้นเพราะเวลา sleep ลดลง

.....

.....

.....

.....

```
#1 P_PID: 14179
#2 C_PID: 14180
#3 P_PID: 14179
#4 P_PID: 14179
#5 C_PID: 14180
HELLO CS KMITL
```

รูปที่ 2 แสดงผลลัพธ์จากการรัน Code ที่ 1 ที่ผ่านการแก้ไข

1.3. หากต้องการให้ผลลัพธ์ของโปรแกรมแสดงผลเป็นลำดับ ดังรูปที่ 2 จะต้องแก้ไข Code อย่างไร จงอธิบาย....

เพิ่ม "\n" ในท้ายของ string ใน printf()

1.4. หากต้องการให้โปรแกรมแสดงผลลัพธ์ของข้อความในบรรทัดที่ 14 จำนวน 2 ครั้ง จะต้องแก้ไข Code อย่างไร โดยไม่เพิ่ม printf() Function และไม่ใช้ Loop จงอธิบาย .....

เอา exit(0) ในบรรทัดที่ 8 ออก

2. System Call: exec() Family – Part I

Library Call Name	Argument List	Pass Current Environment Variables	Search PATH automatic?
execl	list	yes	no
execv	array	yes	no
execle	list	no	no
execve	array	no	no
execlp	list	yes	yes
execvp	array	yes	yes

รูปที่ 3 แสดง System Call “exec()” Family และคุณสมบัติ

ทดสอบรัน Code ที่ 2 – แก้ไข Code ให้ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์ และตอบคำถาม 2.1 - 2.4

Code ที่ 2

No.	File Name: osLab5-02.c
1	void main() {
2	printf("PID: %d\n", getpid());
3	printf("HELLO ...\n");
4	execlp("echo", "echo", "... CS", NULL);
5	printf("... KMITL\n");
6	}

2.1. จงแสดงผลลัพธ์จากการรันโปรแกรม .....

PID: 386185 .....

HELLO ... .

... CS .....

.....

.....

2.2. ทดสอบรันโปรแกรมโดย ปิดและเปิด Comment ในบรรทัดที่ 4 ผลลัพธ์ของการรันโปรแกรมจะมีค่าเหมือนหรือต่างกันอย่างไร จงอธิบาย .....

ต่างกันเพราะไม่มี execlp เพื่อ echo ... CS .....

.....

.....

.....

.....

.....

รูปที่ 4 แสดงผลลัพธ์จากการรัน Code ที่ 2 ที่ผ่านการแก้ไข

2.3. หากต้องการให้ผลลัพธ์ของโปรแกรมแสดงผลลัพธ์ดังรูปที่ 4 จะต้องปรับปรุง Code ที่ 2 อย่างไร - เขียน Code ที่ปรับปรุงในช่องสี่เหลี่ยมด้านล่าง

2.4. ในบรรทัดที่ 4 หากต้องการใช้ System Call “execv()” แทน “execlp()” และกำหนดให้เรียกใช้โปรแกรม “ls -a -l” จะต้องเปลี่ยน Code อย่างไร .....
.....
.....
.....

3. System Call: exec() Family – Part II

ทดสอบรัน Code ที่ 3 – แก้ไข Code ให้ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์ และตอบคำถาม 3.1 - 3.4

Code ที่ 3

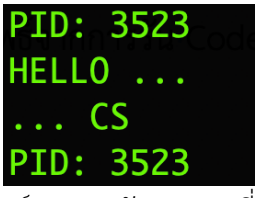
No.	File Name: osLab5-03-1.c
1	void main() {
2	printf("PID: %d\n", getpid());
3	char *argv[] = {"myProgram", "CS", "KMITL", NULL};
4	execvp("myProgram", argv);
5	printf("Bye Bye ...\n");
6	}
No.	File Name: osLab5-03-2.c
1	void main(int argc, char *argv[]) {
2	printf("HELLO ...\n... %s\n", argv[2]);
3	printf("PID: %d\n", getpid());
4	}

หมายเหตุ: - Compile ไฟล์ osLab5-03-2.c โดยกำหนดชื่อโปรแกรมเป็น “myProgram”
- เพิ่ม myProgram ให้อยู่ในตัวแปร $PATH ของ Linux OS

3.1. จงแสดงผลลัพธ์จากการรันโปรแกรม.....
.....
.....
.....
.....

3.2. ผลลัพธ์จากการรันโปรแกรมค่า PID เหมือนกันหรือต่างกันเพราะอะไร จงอธิบาย .....
.....
.....
.....
.....

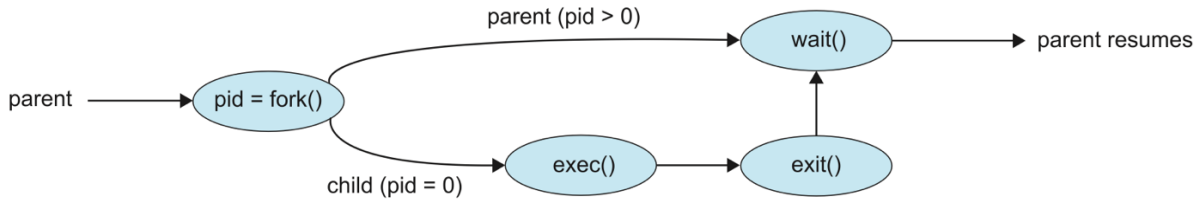
3.3. ผลลัพธ์จากการรันโปรแกรม ไม่แสดงผลของคำสั่งในบรรทัดที่ 5 ในไฟล์ “osLab5-03-1.c” เพราะอะไร จงอธิบาย .....
.....
.....
.....



รูปที่ 5 แสดงผลลัพธ์จากการรัน Code ที่ 3 ที่ผ่านการแก้ไข

3.4. หากต้องการให้ผลลัพธ์ของโปรแกรมแสดงผลลัพธ์ดังรูปที่ 5 จะต้องปรับปรุง Code ที่ 3 ในไฟล์ชื่ออะไร และที่บรรทัดใด แก้ไขอย่างไร .....
.....
.....
.....
.....

4. System Call: fork() and exec() Family - Part I



รูปที่ 5 การทำงานร่วมกันระหว่าง fork() และ exec() และประสานจังหวะผ่าน wait() และ exit()

ทดสอบรัน Code ที่ 4 – แก้ไข Code ให้ Compile ได้ ทดสอบรัน ศึกษาผลลัพธ์ และตอบคำถาม 4.1 - 4.4

Code ที่ 4

No.	File Name: osLab5-04.c
1	void main(int argc, char *argv[]) {
2	pid_t c_pid, gc_pid;
3	c_pid = fork();
4	if (c_pid < 0) {
5	printf("Fork Failed\n");
6	} else if (c_pid == 0) {
7	gc_pid = fork();
8	if (gc_pid == 0) {
9	printf("Number: 1\n");
10	exit(0);
11	}
12	execlp("whoami", argv[1], NULL);
13	printf("Number: 2\n");
14	} else {
15	printf("Number: 3\n");
16	wait(NULL);
17	printf("Number: 4\n");
18	exit(0);
19	}
20	}

4.1. จงแสดงผลลัพธ์จากการรันโปรแกรม.....
.....
.....
.....
.....
4.2. หากต้องการให้บรรทัดที่ 12 ทำงานจะต้องส่งคำสั่ง รันโปรแกรมอย่างไร .....
.....
.....
4.3. หากต้องการให้บรรทัดที่ 12 ทำงาน 2 ครั้งจะต้อง Comment บรรทัดใดของ Code .....
4.4. บรรทัดที่ 12 หากเปลี่ยนการเรียกใช้ System Call “execlp()” เป็น “execl()” ทำการ Compile Code และทดสอบรันโปรแกรม จากผลลัพธ์จะพบว่า Code บรรทัดที่ 13 ถูกประมวลผล จงอธิบายถึงเหตุผลดังกล่าว .....
.....
.....

5. System Call: fork() and exec() Family - Part II

5.1. จงเติม Code ที่ 5 ให้สมบูรณ์เพื่อสร้างโปรแกรมจำลองการทำงานของ Shell แบบมีเมนูให้เลือกเรียกคำสั่งใช้งาน โดยใช้หลักการของ System Call: fork(), wait(), exit(), และ exec() Family ในการพัฒนาโปรแกรม โดยมีข้อกำหนดดังนี้

- มีหน้าแสดง Menu ดังรูปที่ 6
- มีการรับข้อมูลนำเข้าจากผู้ใช้งานผ่าน Keyboard
- ออกจากโปรแกรมโดยกด “q”

```
Select a letter to run a program
'a' run --> uname -r
'b' run --> ls -l
'c' run --> ps -f
'd' run --> df -h
Exit a program press 'q'
Enter a letter to run a program:
```

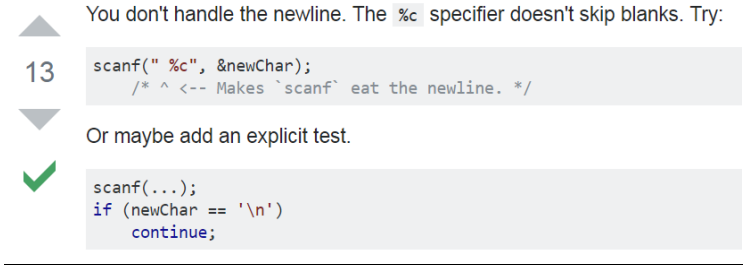
รูปที่ 6 แสดงหน้า Menu ของโปรแกรม ใน Code ที่ 5

หลักการทำงาน – ผู้ใช้งานเลือกใส่ข้อมูลนำเข้าตามตัวอักษรที่กำหนด โปรแกรมรับตัวอักษรแล้วตรวจสอบค่า ตรงกับเงื่อนไขใด โปรแกรมจะทำการสร้าง Process ใหม่โดยเรียก fork() function จากนั้น Child Process ที่ได้นั้น จะเรียก exec() function เพื่อนำไปใช้เรียกโปรแกรม (คำสั่ง) ที่ได้ระบุไว้ตามเงื่อนไขอีกที – มีการ Loop กลับมายังหน้า Menu ของโปรแกรม หากผู้ใช้งานยังไม่กดตัวอักษร “q” เพื่อออกจากโปรแกรม

Code ที่ 5

No.	File Name: osLab5-05.c
1	int main() {
2	pid_t id;
3	char input = 'z';
4	while (input != 'q') {
5	printf("\nSelect a letter to run a program\n");
6	printf("'a' run --> uname -r\n");
7	printf("'b' run --> ls -l\n");
8	printf("'c' run --> ps -f\n");
9	printf("'d' run --> df -h\n");
10	printf("Exit a program press 'q'\n");
11	printf("Enter a letter to run a program: ");
12	scanf(" %c", &input);
13	
14	**<< Your Code Here>>**
15	
16	}
17	return 0;
	}

หมายเหตุ: scanf() ต้องมี space bar หน้า %c เพื่อตัด newline ออก



เขียน หรือ แปะ Source Code ของ Lab ข้อที่ 5 ในพื้นที่ว่างด้านล่าง