05506007 Operating Systems (1/2568)

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

รหัสนักศึกษา.....ชื่อ-สกุล.....

ปฏิบัติการ ครั้งที่ 4 – System Call fork(), wait(), and exit()

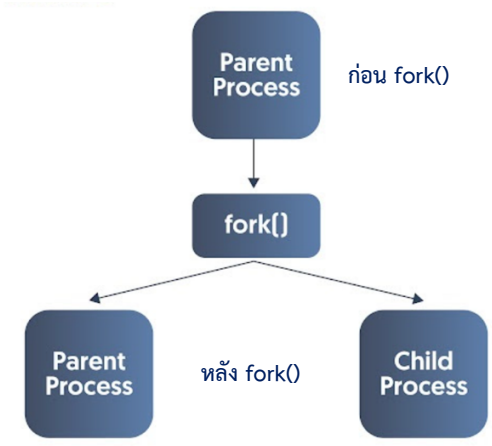
วัตถุประสงค์ของการปฏิบัติการ

- อธิบายการทำงานของ System Call fork(), wait(), และ exit() ได้
- ใช้งาน System Call fork(), wait(), และ exit() ได้

1. System Call “fork()”

System Call คือ Library สำหรับนักร้องขอ OS Kernel ให้บริการตามความสามารถของ System Call นั้น ๆ

fork() เป็น System Call สำหรับสร้าง Process ใหม่ โดยเป็นการ Duplicate (Copy) Process และมีความสัมพันธ์ของ Process ที่เรียก fork() เป็น Parent Process และ Process ที่ถูก fork() เป็น Child Process แสดงความสัมพันธ์ดังรูปที่ 1



รูปที่ 1 แสดงการสร้าง Process ของ fork()

อาจารย์: อนุพงษ์ บรรจงการ

31/07/2568

ศึกษา Code ที่ 1 และตอบคำถามต่อไปนี้

Code ที่ 1

No.	File Name: forkExample01.c
1	#include <stdio.h>
2	#include <unistd.h>
3	#include <sys/types.h>
4	
5	int main() {
6	// fork();
7	pid_t id = getpid();
8	printf("Hello CS KMITL !! : ID is %d\n", id);
9	sleep(30);
10	return 0;
11	}

หมายเหตุ: getpid() ใช้สำหรับเรียกค่าของ Process ID

1.1 หากต้องการ Compile โปรแกรมภาษา C “forkExample01.c” โดยกำหนดชื่อของโปรแกรมหลังการ Compile เป็น “forkExample01” จะต้องใช้คำสั่งอะไร .....

.....

1.2 ทำการรันโปรแกรม “forkExample01” ด้วยคำสั่ง “./forkExample01” จากนั้นเปิด Terminal ขึ้นมาใหม่พร้อมกับรันคำสั่ง “ps -eo pid,user,comm= | grep forkExample01” สังเกตผลลัพธ์ของค่า pid ตรงกันกับผลลัพธ์ของ id จากโปรแกรม “forkExample01” หรือไม่ และมีตัวเลขเป็นเท่าไร .....

.....

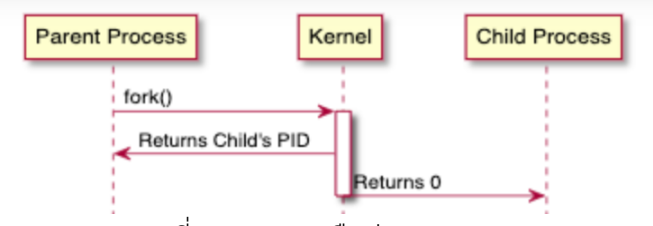
1.3 จาก Code ที่ 1 หาก uncomment บรรทัดที่ 6 ทำการ recompile code แล้วทำการรันโปรแกรม จะได้ผลลัพธ์จากการรันกี่บรรทัด และมีเลขของ id เหมือนกันหรือไม่ เพราะอะไร .....

.....

.....

1.4 หากเพิ่ม “fork()” เข้าไประหว่างบรรทัดที่ 6 กับ 7 อีกสองคำสั่ง ทำการ recompile code แล้วทำการรันโปรแกรม จะได้ผลลัพธ์จากการรันกี่บรรทัด .....

การเรียก System Call “fork()” เพื่อสร้าง Process ใหม่นั้น ผลการเรียกจะมีการ Return ค่ากลับมา – โดยถ้า fork() สำเร็จจะส่งค่า 0 ให้ Child Process และส่งค่า pid ของ Child Process ซึ่งจะมีค่ามากกว่า 0 ให้กับ Parent Process แสดงดังรูปที่ 2 และถ้า fork() ไม่สำเร็จจะให้ค่า Return เป็นลบ



รูปที่ 2 แสดงการคืนค่าของ fork()

ศึกษา Code ที่ 2 และตอบคำถามต่อไปนี้

Code ที่ 2

No.	File Name: forkExample02.c
1	#include <stdio.h>
2	#include <unistd.h>
3	#include <sys/types.h>
4	
5	int main() {
6	int i = 0;
7	pid_t id = fork();
8	if (id > 0) {
9	i += 1;
10	printf("I'm parent\t i = %d\n", i);
11	} else {
12	i += 2;
13	printf("I'm child\t i = %d\n\n", i);
14	}
15	return 0;
16	}

1.5 จาก Code ที่ 2 หาก Compile Code โดยกำหนดชื่อของโปรแกรมเป็น “forkExample02” จากนั้นทำการรันโปรแกรม นักศึกษาคิดว่าโปรแกรมจะให้ผลลัพธ์กี่บรรทัด .....

1.6 จากการรันโปรแกรมสังเกตุรูปแบบของผลลัพธ์มีกี่รูปแบบ อะไรบ้าง (ทดสอบรันโปรแกรมหลาย ๆ รอบด้วยการใช้ shell script ดังนี้ “for i in {1..100}; do ./forkExample02; done”) .....

.....

.....

.....

.....

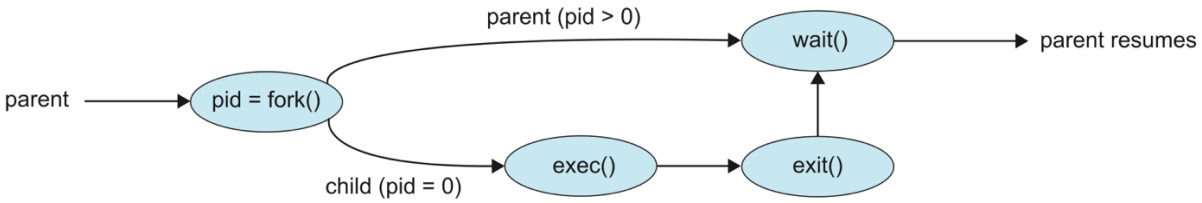
1.7 เติม Code ในช่องว่าง (.....) ของ Code ที่ 3 จากนั้นให้ Compile Code แล้วรันเพื่อศึกษาผลลัพธ์ของโปรแกรม ทบทวนและทำความเข้าใจเกี่ยวกับ Parent และ Child Process ของการ fork()

Code ที่ 3

No.	File Name: forkExample03.c
1	#include <stdio.h>
2	#include .....
3	#include <sys/types.h>
4	
5	int main() {
6	..... id;
7	printf("<pid: %d>\n\n", getpid());
8	id = ...;
9	if (.....) {
10	fprintf(stderr, "fork() failed\n");
11	} else if (.....) {
12	printf("I'm child\tfork return id = %d\t<pid: %d>\n", id, ...);
13	} else {
14	printf("I'm parent\tfork return id = %d\t<pid: %d>\n", id, ...);
15	}
16	return 0;
17	}

2. System Call “fork(), wait(), and exit()”

จาก Lab ข้อที่ 1 สามารถแยก Code ส่วนของ Parent กับ Child Process ผ่านค่า Return ของ fork() แต่ยังไม่ได้กำหนดจังหวะการทำงานของ Parent กับ Child Process ใน Lab ข้อที่ 2 นี้จะใช้ System Call “wait()” และ “exit()” เพื่อกำหนดจังหวะการทำงานของ Parent กับ Child Process โดยเมื่อ Process ทำการ fork() จะกำหนดให้ Parent Process รอ Child Process ทำงานให้เสร็จก่อนแล้วจึงค่อยกลับมาทำงานในส่วนของ Parent Process ดังรูปที่ 3.



รูปที่ 3 การประสานจังหวะทำงานของ Parent และ Child Process ผ่าน wait() และ exit()

ศึกษา Code ที่ 4 แล้วตอบคำถาม

Code ที่ 4

No.	File Name: forkWaitExample01.c
1	#include <stdio.h>
2	#include <unistd.h>
3	#include <sys/types.h>
4	#include <sys/wait.h>
5	
6	int main() {
7	pid_t id;
8	int i = 0;
9	id = fork();
10	if (id > 0) {
11	// wait(NULL);
12	i += 1;
13	printf("I'm parent, i = %d\n", i);
14	} else {
15	i += 2;
16	printf("I'm child, i = %d\n", i);
17	// exit(0);
18	}
19	printf("my i = %d\n", i);
20	return 0;
21	}

2.1 จาก Code ที่ 4 หากทำการ Compile Code โดยกำหนดชื่อโปรแกรม “forkWaitExample01” จากนั้นทำการรันโปรแกรม คำถาม OS จะรัน Process ของ Parent หรือ Child ก่อน .....

.....

.....

2.2 จาก Code ที่ 4 หาก uncomment บรรทัดที่ 11 recompile และทำการรันโปรแกรม จงแสดงผลลัพธ์ที่ได้จากการรัน .....

.....

.....

.....

.....

.....

2.3 จาก Code ที่ 4 หาก uncomment บรรทัดที่ 11 และ 17 recompile และทำการรันโปรแกรม คำถามเกิดปัญหาขณะ compile หรือไม่ ถ้ามีแก้ไขอย่างไร และหากแก้ไขเรียบร้อยแล้ว ทำการรันโปรแกรม code บรรทัดที่ 19 ทำงานกี่ครั้ง .....

.....

.....

.....

.....

2.4 จาก Code ที่ 5 จงอธิบายความแตกต่างของผลลัพธ์จากการรันโปรแกรมระหว่างการ comment และ uncomment ในบรรทัดที่ 19 ของ Code .....

.....

.....

.....

.....

2.5 จาก Code ที่ 5 หาก Comment ในบรรทัดที่ 15 ของ Code ทำการรันโปรแกรม Code ในบรรทัดที่ 20 จะทำงานกี่ครั้ง .....

.....

2.6 จาก Code ที่ 5 หากเพิ่มจำนวน for loop จากค่า 3 เป็น 6 ทำการรันโปรแกรมโดย comment บรรทัดที่ 15 ไว้ คำถาม Code ในบรรทัดที่ 20 จะทำงานกี่ครั้ง .....

.....

ศึกษา Code ที่ 5 แล้วตอบคำถาม

Code ที่ 5

No.	File Name: forkWaitExample02.c
1	#include <stdio.h>
2	#include <stdlib.h>
3	#include <unistd.h>
4	#include <sys/types.h>
5	#include <sys/wait.h>
6	
7	int main() {
8	pid_t id;
9	int num = 10;
10	int i;
11	for (i = 0; i < 3; i++) {
12	id = fork();
13	if (id == 0) {
14	printf("I'm Child, i = %d\n", i);
15	exit(0);
16	printf("Should not be executed\n");
17	}
18	}
19	while(wait(NULL) != -1);
20	printf("Parent number = %d\n", num);
21	return 0;
22	}

Code ที่ 6

No.	File Name: forkWaitExample03.c
1	...
2	int main() {
3	pid_t id;
4	int i = 0;
5	printf("Enter a positive number: ");
6	scanf("%d", &num);
7	
8	if (num <= 0) {
9	printf("You did not enter a positive number\n");

```
10          exit(1);
11      }
12
13      if ((id = fork()) > 0) {
14          for (i = 1; i <= num; i++) {
15              sum += i;
16          }
17          wait(NULL);
18          printf("I'm parent, my sum = %d\n", sum);
19      } else {
20          for (i = 1; i <= 2 * num; i++) {
21              sum += i;
22          }
23          printf("I'm child, my sum = %d\n", sum);
24          exit(0);
25      }
26      return 0;
27  }
```

2.7 จาก Code ที่ 6 ปรับปรุงแก้ไข Code ให้สามารถ Compile ได้ จากนั้นทดสอบรันโดยใส่ค่าตัวเลข input เป็น 5 คำถาม Parent Process หรือ Child Process ได้ทำงานก่อน และค่าของตัวแปร “sum” ของแต่ละ Process มีค่าเป็นเท่าไร .....

.....

.....

2.8 จาก Code ที่ 7 จงเติม Code ให้สมบูรณ์ โดยโปรแกรมต้องการเงื่อนไขดังนี้

- Parent Process วน Loop for เพื่อ fork Child Process จำนวน 5 Processes
- Child Process ที่ถูก fork จาก index ของ Loop for ที่เป็นเลขคู่ (0, 2, 4) จะวน Loop เพื่อ fork Child Process ของตนเองอีก 3 Processes ส่วน index ของ Loop for ที่เป็นเลขคี่ (1, 3) จะวน Loop เพื่อ fork Child Process ของตนเองอีก 4 Processes
- โดยที่ Process แต่ละ Process จะทำงานโดยพิมพ์ข้อความแสดงตัวออกมาหนึ่งบรรทัดเท่านั้น

Code ที่ 7

No.	File Name: forkWaitExample04.c
1	int main() {

```
2       pid_t id, sub_id;
3       int i, j;
4       printf("Only parent before fork\n");
5       for (i = 0; i < 5; i++) {
6               id = fork();
7               if (id == 0) {
8                       if ((i % 2) == 0) {
9                               printf("I'm child no.%d\n", i);
10                              int num_gc = .....;
11                              for (j = 0; j < num_gc; j++) {
12                                      .....;
13                                      if (sub_id == 0) {
14                                         printf("I'm gc num:%d of even child no.%d\n", j, i);
15                                         .....;
16                                      }
17                              }
18                              .....;
19                              exit(0);
20                      } else {
21                              printf("I'm child no.%d\n", i);
22                              int num_gc = .....;
23                              for (j = 0; j < num_gc; j++) {
24                                      sub_id = fork();
25                                      if (.....) {
26                                         printf("I'm gc num:%d of odd child no.%d\n", j, i);
27                                         .....;
28                                      }
29                              }
30                              while(wait(NULL) != -1);
31                              .....;
32                      }
33              }
34      }
35      while(wait(NULL) != -1);
36      return 0;
37  }
```