มหาวิทยาลัยสงขลานครินทร์

คณะวิศวกรรมศาสตร์

สอบปลายภาค ประจำภาคการศึกษาที่ 1 ปีการศึกษา 2556

วันที่ 11 ตุลาคม 2556 เวลา 0900-1200

วิชา 242-533 Advanced Unix Network Programming ห้อง S817

- ข้อสอบมีทั้งหมด 6 หน้า รวมปก
- ข้อสอบมี 2 part โดย Part 1 มีคำถาม 12 ข้อ, Part II ให้เขียนโปรแกรม 1 โปรแกรม ให้ทำทุกข้อ
- อนุญาตให้นำเครื่องคำนวณและเอกสารเข้าห้องสอบได้

ทุจริตในการสอบ โทษขั้นต่ำคือปรับตกในรายวิชาที่ทุจริต และพักการเรียน 1 ภาคการศึกษา

Part I. จงตอบคำถามต่อไปนี้

1.) จากส่วนของโปรแกรมต่อไปนี้ จงวาดรูปของ process tree ที่เกิดจากการใช้คำสั่ง fork() แบบเดียวกับผลลัพธ์ที่ได้จากคำสั่งของ pstree กำหนดให้ทุกครั้งของการเรียกใช้คำสั่ง fork() ไม่เกิด error

1.1

```
main() {
...
fork();
fork();
fork();
exit(0);
}
```

1.2

```
main() {
int i;
...
for (i=0;i<3;i++) {
      int pid = fork();
      if (pid == 0) exit(0;
}
```

1.3

```
main() {
int i;
...
for (i=0;i<3;i++) {
      int pid = fork();
      if (pid != 0) exit(0;
}
```

2.) จงอธิบายผลของคำสั่ง wait() ที่มีต่อทั้ง parent process และ child process ถ้ามีการเรียกใช้คำสั่งนี้ทั้งในกรณีที่ parent process terminate ก่อน หรือ หลัง child process

3.) จงอธิบายข้อจำกัดของการใช้งาน popen() ในการสื่อสารระหว่าง process

4.) การใช้งาน pipe สำหรับการสื่อสารแบบสองทาง ระหว่าง parent process และ child process จะต้องมีกระบวนการในการสร้าง pipe และ fork process อย่างไร จึงจะสามารถใช้ pipe ในการสื่อสารแบบ 2 ทางได้อย่างถูกต้อง ให้ยกตัวอย่างส่วนของโปรแกรม พร้อมคำอธิบาย

5.) ในการใช้งาน FIFO เพื่อสำหรับการสื่อสารระหว่าง process สามารถสร้างโดยวิธีการใดบ้าง ให้เขียนตัวอย่างของคำสั่ง หรือส่วนของโปรแกรม ประกอบคำอธิบาย

6.) การส่ง signal ระหว่าง process สามารถส่งได้โดยวิธีการใดบ้าง ให้ยกตัวอย่างคำสั่ง และ ส่วนของโปรแกรมประกอบคำอธิบาย

7.) อธิบายผลของการกำหนด signal ในส่วนของโปรแกรมหนึ่งดังต่อไปนี้ ว่าจะส่งผลอย่างไรบ้าง เมื่อโปรแกรมได้รับ signal นั้นๆ

```
signal(SIGTERM, SIG_DFL);
signal(SIGUSR1, SIG_DFL);
signal(SIGHUB, SIG_IGN);
signal(SIGKILL, SIG_IGN);
```

8.) จงแสดงเขียนส่วนของโปรแกรม สำหรับใช้เป็น signal handler function สำหรับนับจำนวนครั้งที่ process ได้รับ signal TERM (จาก process ใดๆ) และ เมื่อได้รับ signal USR1 แล้วจะพิมพ์จำนวนครั้งที่นับนั้นออกมาทาง terminal (อาจจะแยกเป็น 2 ฟังก์ชัน หรือ ฟังก์ชันเดียว สามารถทำได้ 2 หน้าที่ก็ได้)

9.) จงแสดงวิธีการกำหนด process ให้ใช้งาน handler ทั้งสองในข้อ 8 โดยการใช้ sigaction()

10.) ในการใช้ shared memory สำหรับการส่งข้อมูลระหว่าง process จะมีข้อแตกต่างจากการใช้ pipe หรือ fifo อย่างไร จงอธิบาย

11.) จงแสดงส่วนของโปรแกรม ซึ่งใช้ในการเตรียม message queue เพื่อใช้ในการสื่อสารระหว่าง parent กับ child process

12.) จงอธิบายการใช้งาน semaphore ว่าจะมีประโยชน์สำหรับงานในลักษณะใด

Part II. เขียนโปรแกรมภาษา C สำหรับใช้งานบนระบบปฏิบัติการแบบ Unix

จากตัวอย่างของโปรแกรมภาษา C ต่อไปนี้ เป็นโปรแกรมสำหรับการเล่นเกม ox แบบมีผู้เล่น 2 คน โดยรับอินพุทเป็นตำแหน่งเป็น coordinate x,y ผ่านทาง standard input และแสดงผลการเล่นทีละขั้นทาง standard output เนื่องจากเป็นเกมที่ผู้เล่นทั้งสองต้องผลัดกันใช้คีย์บอร์ดในการป้อนอินพุทในการเล่นเกม ทำให้ไม่สะดวกต่อการเล่น ให้ดัดแปลงแก้ไขโปรแกรมตัวอย่าง เพื่อแยกโปรเซสของการรับอินพุท ออกจากโปรแกรมหลัก และให้โปรเซสทั้งสองติดต่อกันผ่านทาง interprocess communication โดยเลือกใช้วิธีการสื่อสารที่เหมาะสม จุดมุ่งหมายเพื่อให้ผู้เล่นทั้งสอง สามารถเล่นเกม ox ด้วยกันได้ โดยไม่จำเป็นจะต้องใช้ คีย์บอร์ด และ จอภาพชุดเดียวกัน

อธิบายหลักการที่ใช้ และแสดงส่วนขอ code ที่แก้ไข เพื่อให้ทำงานตามที่อธิบายมาได้ โปรแกรมไม่จำเป็นจะต้องครบสมบูรณ์ แต่ควรที่จะครอบคลุมองค์ประกอบที่สำคัญในส่วนของการสื่อสารระหว่างโปรเซสทั้งหมด

ตัวอย่างโปรแกรม ox.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char    table[3][3];

void    init_table(void) {
    int i,j;
    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            table[i][j]=' ';
}

void    draw_table(void) {
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            printf("[%c]", table[i][j]);
        }
        printf("\n");
    }
    printf("---------\n");
}
```

```
void    take(char c) {
        int x,y;
        while (1) {
            printf("%c turn -> input y x : ", c);
            scanf("%d %d", &x, &y);
            if ((x<0 || x>2) || (y<0 || y>2)) {
                printf("out of range\n");
                continue;
            }
            if (table[x][y] != ' ') {
                printf("[%d,%d] = '%c', please choose again\n",
                        x,y,table[x][y]);
                continue;
            }
            table[x][y] = c;
            break;
        }
}

void    xturn() {
        take('x');
}

void    oturn() {
        take('o');
}

void checkrow(char c) {
    int i,j;
    for (i=0;i<3;i++) {
        int count = 0;
        for (j=0;j<3;j++) {
            if (table[i][j]==c)
                count++;
        }
        if (count == 3) {
            printf("%c win row=%d\n", c, i);
            exit(0);
        }
    }
}
```

```
void checkcol(char c) {
    int i,j;
    for (i=0;i<3;i++) {
        int count = 0;
        for (j=0;j<3;j++) {
            if (table[j][i]==c)
                count++;
        }
        if (count == 3) {
            printf("%c win col=%d\n", c, i);
            exit(0);
        }
    }
}

void checkcross(char c) {
    if ((table[0][0] == c) &&
        (table[1][1] == c) &&
        (table[2][2] == c)) {
            printf("%c win cross backward\n", c);
            exit(0);
    }
    if ((table[0][2] == c) &&
        (table[1][1] == c) &&
        (table[2][0] == c)) {
            printf("%c win cross forward\n", c);
            exit(0);
    }
}

void    check() {
    checkrow('x');
    checkcol('x');
    checkcross('x');
    checkrow('o');
    checkcol('o');
    checkcross('o');
}

int main(void) {
    init_table();
    while(1) {
        draw_table();   xturn();    check();
        draw_table();   oturn();    check();
    }
}
```