**Activity 3: Process Concept**

**สมาชิก**

|  |  |
| --- | --- |
| **ชื่อ-นามสกุล** | **เลขประจำตัวนิสิต** |
| นายเนติภัทร โพธิพันธ์ | 6631331621 |
| นายวรลภย์ ศรีชัยนนท์ | 6632200221 |
| นายสิปปภาส ชวานนท์ | 6630333721 |

1. **เขียนโปรแกรมสร้างโปรเซส 3 ระดับ โดยที่โปรเซสแต่ละระดับต้องบอกว่า ตัวเองเป็น parent, child หรือ grandchild แล้วแสดงค่า pid ของตนเอง และค่า ppid ด้วยโดยใช้ fork(), getpid() และ getppid()**

**#include<stdio.h>**

**#include<unistd.h>**

**#include<sys/wait.h>**

**int main() {**

**int pid1, pid2;**

**pid1 = fork();**

**if(pid1 == 0) {**

**// Child Process**

**printf("I am the child process. My PID is %d ", getpid());**

**printf("and my parent's PID is %d\n", getppid());**

**pid2 = fork();**

**if(pid2 == 0) {**

**// Grandchild Process**

**printf("I am the grandchild process. My PID is %d ", getpid());**

**printf("and my parent's PID is %d\n", getppid());**

**}**

**else {**

**waitpid(pid2, NULL, 0); // Child waits for its grandchild**

**}**

**}**

**else {**

**// Parent Process**

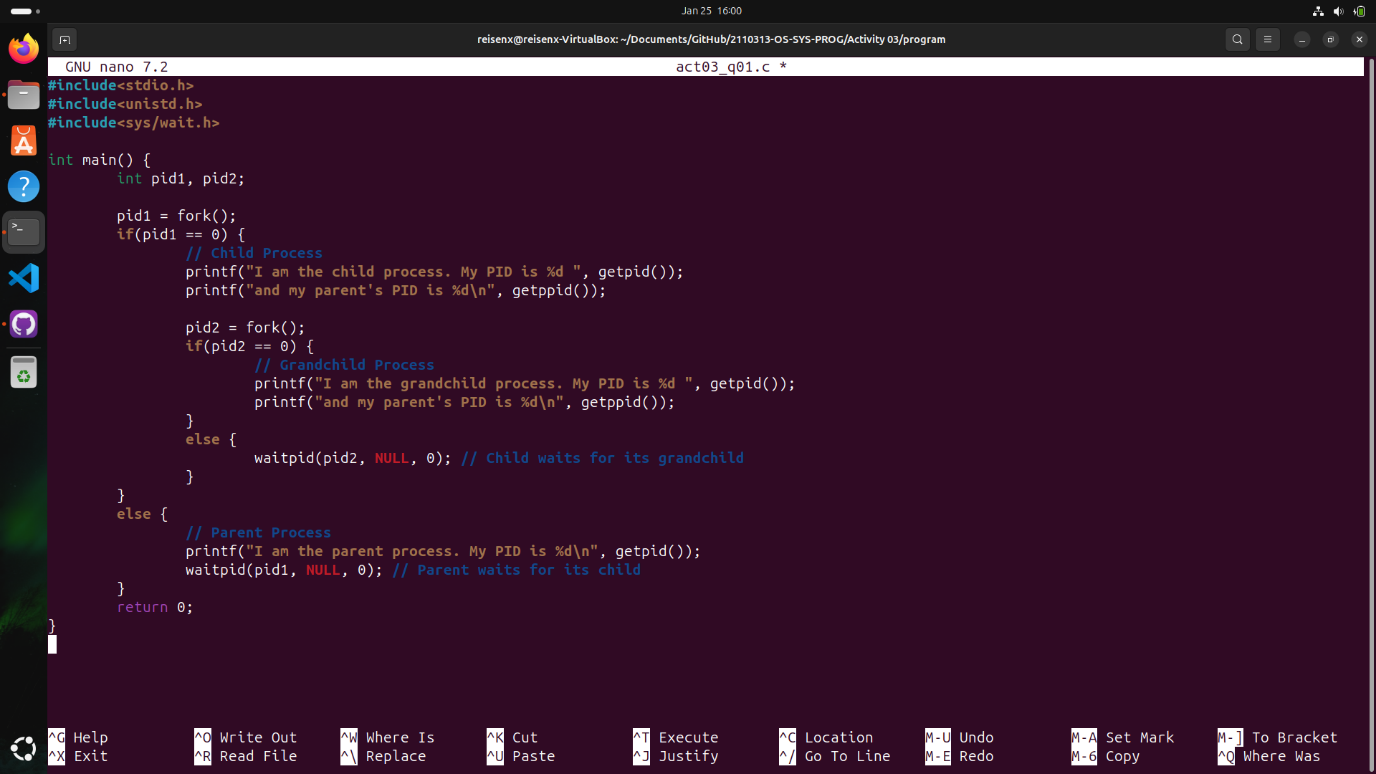
**printf("I am the parent process. My PID is %d\n", getpid());**

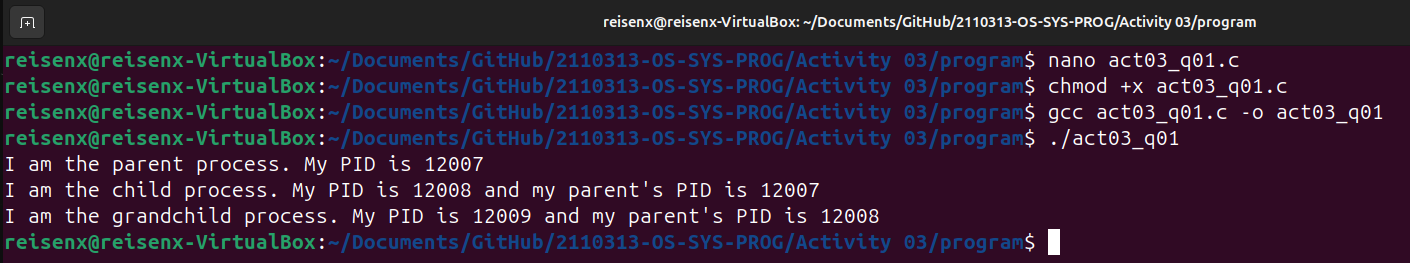
**waitpid(pid1, NULL, 0); // Parent waits for its child**

**}**

**return 0;**

**}**

****

****

1. **โปรแกรมต่อไปนี้ทำงานสลับระหว่างการคำนวณและการไม่ทำอะไรเลย (sleep)**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**#include<signal.h>**

**int compute\_period = 5;**

**int sleep\_period = 5;**

**int i;**

**/\* what to do when alarm is on \*/**

**void on\_alarm (int signal) {**

**printf("Sleep\n");**

**sleep(sleep\_period);**

**printf("Wake up\n");**

**/\* activate alarm again \*/**

**alarm(compute\_period);**

**}**

**main(int argc, char\* argv[]) {**

**int i;**

**if(argc != 3) {**

**printf("Usage: infinite <compute-period><sleepperiod>\n");**

**exit(0);**

**}**

**else {**

**compute\_period = atoi(argv[1]);**

**sleep\_period = atoi(argv[2]);**

**}**

**/\* on\_alarm is signal handler for SINGLARM \*/**

**signal(SIGALRM, on\_alarm);**

**/\* activate alarm \*/**

**alarm(compute\_period);**

**/\* compute infinitely but can be interrupted by alarm \*/**

**for(i = 0; ;i++) {**

**if(i == 0) printf("computing\n");**

**}**

**}**

**ให้แก้ไขโปรแกรมโดยเปลี่ยนจากการใช้ argument เป็นการถามค่า compute\_period และ sleep\_period จากแป้นพิมพ์**

**Solution Code**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h>**

**#include<signal.h>**

**int compute\_period = 5;**

**int sleep\_period = 5;**

**/\* what to do when alarm is on \*/**

**void on\_alarm (int signal) {**

**printf("Sleep\n");**

**sleep(sleep\_period);**

**printf("Wake up\n");**

**/\* activate alarm again \*/**

**alarm(compute\_period);**

**}**

**int main() {**

**/\* Input from a keyboard instead of using arguments \*/**

**printf("Enter compute period : \n");**

**scanf("%d", &compute\_period);**

**printf("Enter sleep period : \n");**

**scanf("%d", &sleep\_period);**

**/\* on\_alarm is signal handler for SINGLARM \*/**

**signal(SIGALRM, on\_alarm);**

**/\* activate alarm \*/**

**alarm(compute\_period);**

**/\* compute infinitely but can be interrupted by alarm \*/**

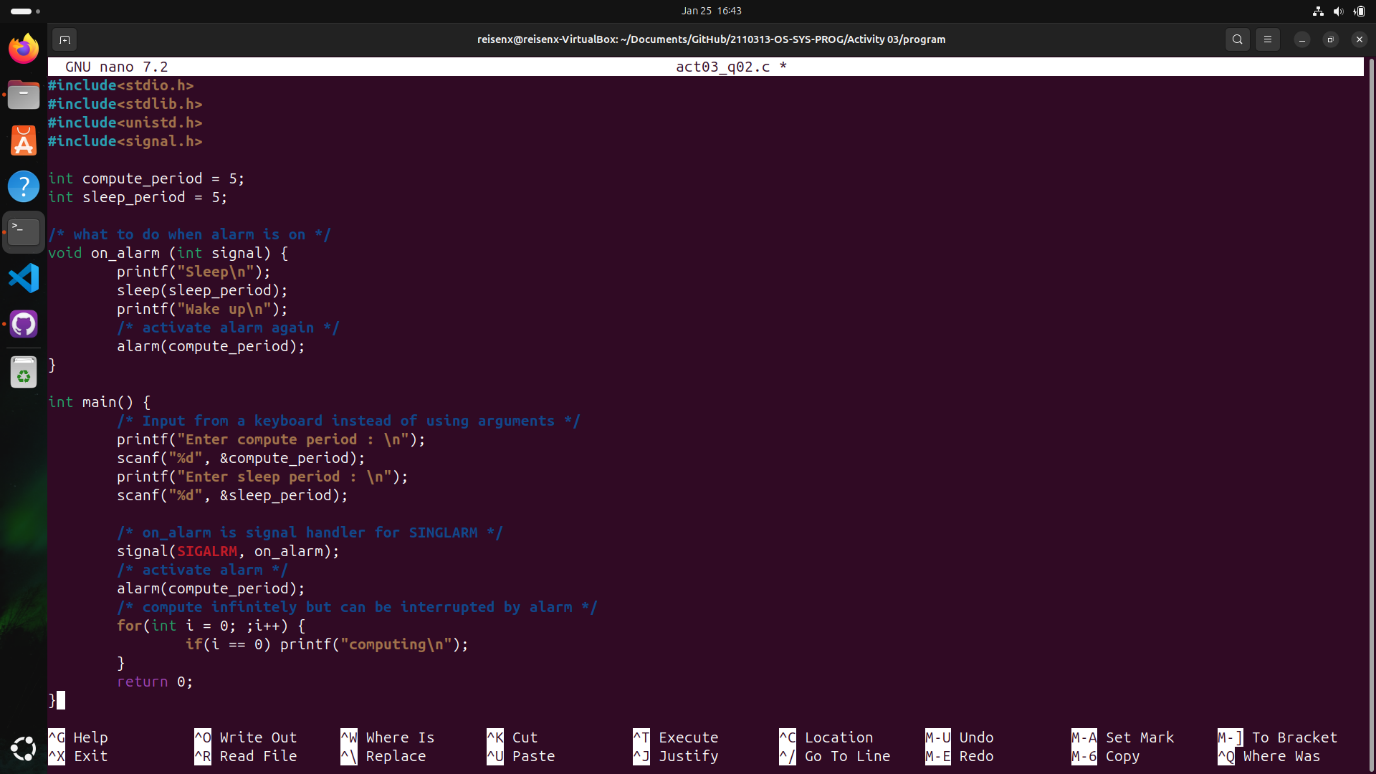
**for(int i = 0; ;i++) {**

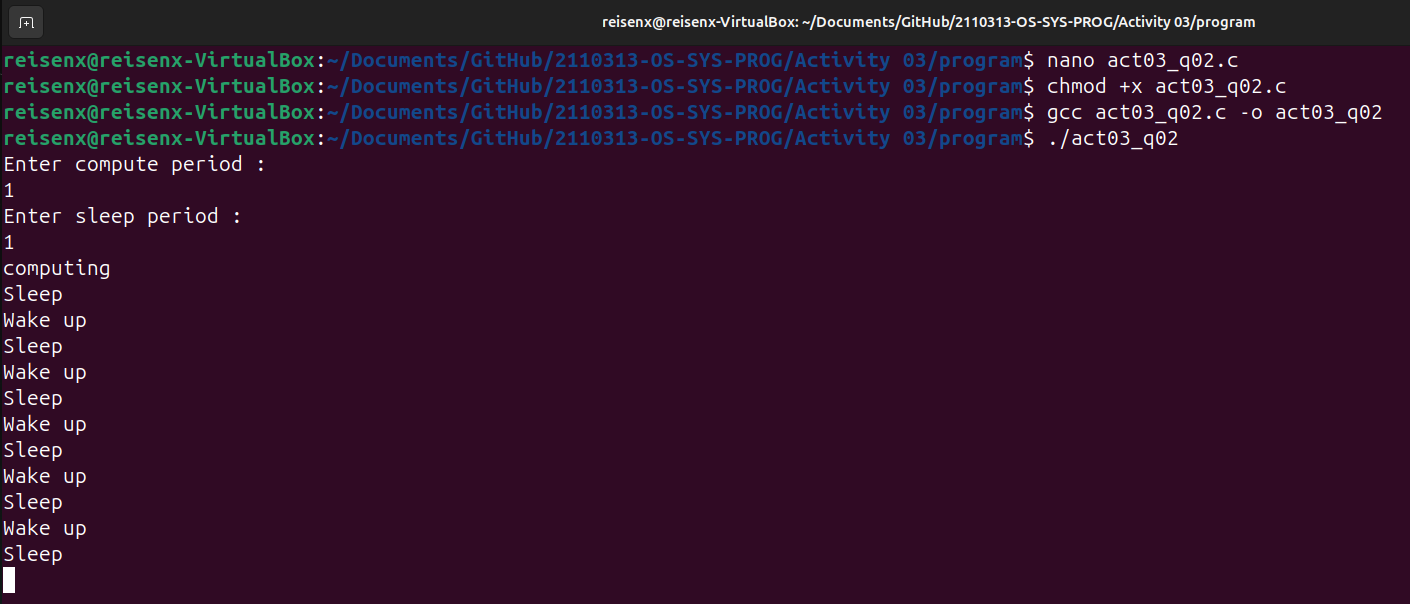
**if(i == 0) printf("computing\n");**

**}**

**return 0;**

**}**

****

****

1. **แก้ไขโปรแกรมข้างล่างนี้ โดยให้กลับลำดับของข้อความที่พิมพ์ออกมา นั่นคือโปรเซสที่ถูกสร้างขึ้นหลังสุด (pid มากที่สุด) จะพิมพ์ข้อความออกมาแรกสุด และโปรเซสที่ถูกสร้างขึ้นแรกสุดจะพิมพ์ข้อความออกมาหลังสุด**

**#include<stdio.h>**

**#include<unistd.h>**

**#include<sys/types.h>**

**main() {**

**int i;**

**int n;**

**pid\_t childpid;**

**n = 4;**

**for(i = 0; i < n; ++i) {**

**childpid = fork();**

**if(childpid > 0) {**

**break;**

**}**

**}**

**printf("This process %ld with parent %ld\n", (long) getpid(), (long) getppid());**

    wait(0);

}

**Solution Code**

**#include<stdio.h>**

**#include<unistd.h>**

**#include<sys/types.h>**

**#include<sys/wait.h>**

**int main() {**

**int n = 4;**

**pid\_t childpid;**

**for(int i = 0; i < n; ++i) {**

**childpid = fork();**

**// Parent process**

**if(childpid > 0) {**

**break;**

**}**

**}**

**// Parent waits for its child to finish**

**wait(0);**

**printf("This process %ld ", (long) getpid());**

**printf("with parent %ld \n", (long) getppid());**

**return 0;**

**}**



