

Activity 4: Simple Shell

สมาชิก

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

1. โปรแกรมข้างล่างแสดงการใช้ fork() และ execvp()

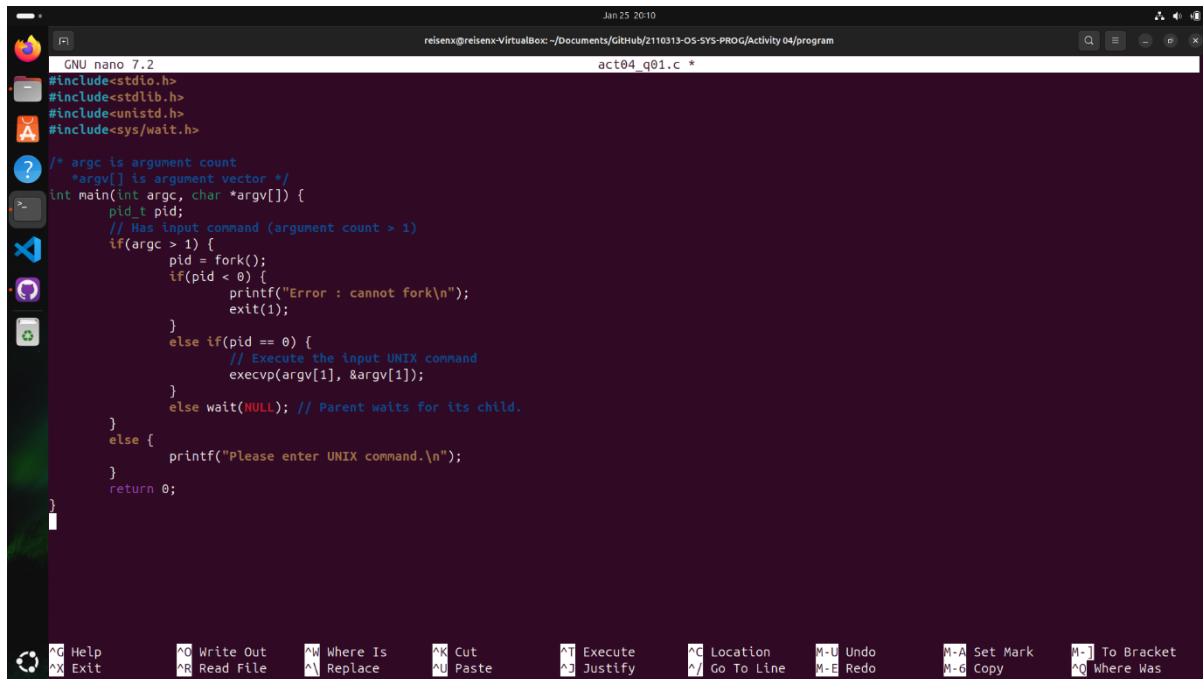
โปรเซสแม่สร้างโปรเซสลูก จากนั้นแม่คอยให้ลูกเรียก execvp() ทำงาน cal 3 2021

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

int main() {
    pid_t pid;
    char *av[] = {"cal", "3", "2021", (char *)0};

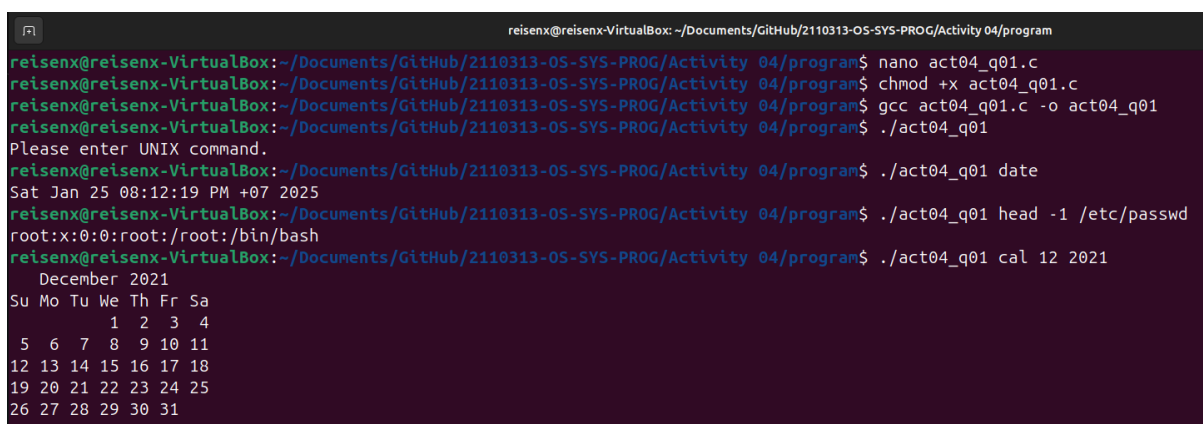
    pid = fork();
    if(pid < 0) {
        printf("Error : cannot fork\n");
        exit(1);
    }
    else if(pid == 0) {
        execvp("cal", av);
    }
    else {
        wait(NULL);
        return 0;
    }
}
```

ขอให้ผลิตดัดแปลงโปรแกรมข้างบนให้รับ argument เป็นคำสั่ง LINUX แล้วทำงานคำสั่ง LINUX ตามที่ป้อน จากนั้นหยุดทำงาน
ถ้าผู้ใช้ไม่ป้อน argument โปรแกรมจะบอกให้ผู้ใช้ป้อนคำสั่งและหยุดทำงาน



```
GNU nano 7.2 act04_q01.c *
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>

/* argc is argument count
 * argv[] is argument vector */
int main(int argc, char *argv[]) {
    pid_t pid;
    // Has input command (argument count > 1)
    if(argc > 1) {
        pid = fork();
        if(pid < 0) {
            printf("Error : cannot fork\n");
            exit(1);
        }
        else if(pid == 0) {
            // Execute the input UNIX command
            execvp(argv[1], &argv[1]);
        }
        else wait(NULL); // Parent waits for its child.
    }
    else {
        printf("Please enter UNIX command.\n");
    }
    return 0;
}
```



```
reisenx@reisenx-VirtualBox: ~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ nano act04_q01.c
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ chmod +x act04_q01.c
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ gcc act04_q01.c -o act04_q01
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ ./act04_q01
Please enter UNIX command.
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ ./act04_q01 date
Sat Jan 25 08:12:19 PM +07 2025
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ ./act04_q01 head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-OS-SYS-PROG/Activity 04/program$ ./act04_q01 cal 12 2021
December 2021
Su Mo Tu We Th Fr Sa
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

2. พิจารณาโปรแกรมข้างล่าง

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

int main() {
    /* the program loops until exit */
    int run = 1;

    while(run) {
        printf("mysh >");
        /*
            After reading user input, do these steps
            1. use tokenize() function to get command
            2. fork a child process
            3. child use execvp() to run command
            4. parent call wait() until user enter "exit"
        */
    }
}

/*
    Split input string into substrings (called tokens)
    which are sequences of contiguous characters separated by any
    of the character

    in the string of accepted delimiters and fill the tokens into
    an array.

    The last element of the array contains a NULL pointer.
    Return number of tokens or -1 if not success

    Example:
    char delim[] = " \t\n";
    char **tokens;
    char string[256];
    int numtokens;
    int i;

    fget(string, 256, stdin);
    numtokens = tokenize(string, delim, &tokens);
    for(i = 0; i < numtokens; i++) {
        printf("%d:%s\n", i, tokens[i]);
    }
*/
```

```

int tokenize(char *string, char *delimiters, char
***arrayOfTokens) {
    char *token;
    int numtokens = 0;
    int i;

    /* skip the beginning delimiters */
    string += strspn(string, delimiters);
    if((token = malloc(strlen(string) + 1)) == NULL) {
        return -1;
    }

    /* count tokens */
    strcpy(token, string);
    numtokens = 0;
    if(strtok(token, delimiters) != NULL) {
        for(numtokens = 1; strtok(NULL, delimiters) != NULL;
numtokens++);
    }

    /* create array of pointers to tokens */
    if((*arrayOfTokens = malloc((numtokens+1) * sizeof(char *)))
== NULL) {
        free(token);
        return -1;
    }

    /* fill pointers to tokens into the array */
    if(numtokens == 0) free(token);
    else {
        strcpy(token, string);
        (*arrayOfTokens)[0] = strtok(token, delimiters);
        for(i = 1; i < numtokens; i++) {
            (*arrayOfTokens)[i] = strtok(NULL, delimiters);
        }
        (*arrayOfTokens)[numtokens] = NULL;
    }
    return numtokens;
}

```

ใส่โค้ดใน main() เพื่อให้ทำงานเป็น shell แบบง่าย โดยที่ shell fork โปรเซสลูกเพื่อให้ลูกทำงานตามคำสั่ง LINUX ตามที่ผู้ใช้ป้อน และแม่ fork ลูกคอยรับคำสั่งใหม่เรื่อย ๆ จนกระทั่งผู้ใช้ป้อน exit จึงจะหยุดทำงาน

Hint: ใช้ fgets() และฟังก์ชัน tokenize() เพื่อรับคำสั่งจากแป้นพิมพ์

```
GNU nano 7.2
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/wait.h>

int tokenize(char *string, char *delimiters, char ***arrayOfTokens) {
    char *token;
    int numtokens = 0;
    /* skip the beginning delimiters */
    string += strspn(string, delimiters);
    if((token = malloc(strlen(string) + 1)) == NULL) {
        return -1;
    }

    /* count tokens */
    strcpy(token, string);
    if(strtok(token, delimiters) != NULL) {
        for(numtokens = 1; strtok(NULL, delimiters) != NULL; numtokens++);
    }

    /* create array of pointers to tokens */
    if((*arrayOfTokens = malloc((numtokens+1) * sizeof(char *))) == NULL) {
        free(token);
        return -1;
    }

    /* fill pointers to tokens into the array */
    if(numtokens == 0) free(token);
    else {
        strcpy(token, string);
        (*arrayOfTokens)[0] = strtok(token, delimiters);
        for(int i = 1; i < numtokens; i++) {
            (*arrayOfTokens)[i] = strtok(NULL, delimiters);
        }
        (*arrayOfTokens)[numtokens] = NULL;
    }
    return numtokens;
}

int main() {
```

```
GNU nano 7.2
int main() {
    /* the program loops until exit */
    int run = 1;
    while(run) {
        printf("mysh >");
        /* variables declaration */
        char string[256];
        char **arrayOfTokens = NULL;
        char delimiters[] = " \t\n";

        /* Input and tokenize to get array of arguments */
        fgets(string, 256, stdin);
        tokenize(string, delimiters, arrayOfTokens);

        /* Exit command */
        if(strcmp(arrayOfTokens[0], "exit") == 0) {
            run = 0;
        }

        /* Normal commands */
        else {
            pid_t pid;
            pid = fork();
            if(pid < 0) {
                printf("Error occurred.\n");
                exit(1);
            }
            else if(pid == 0) {
                /* Execute the command */
                execvp(arrayOfTokens[0], arrayOfTokens);
            }
            else {
                wait(NULL); // parents waits for its child.
            }
        }

        /* Free the allocated memory after execute in each loop */
        free(arrayOfTokens[0]);
        free(arrayOfTokens);
    }
    return 0;
}
```

```
reisenx@reisenx-VirtualBox: ~/Documents/GitHub/2110313-05-SYS-PROG/Activity 04/program

reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-05-SYS-PROG/Activity 04/program$ chmod +x act04_q02.c
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-05-SYS-PROG/Activity 04/program$ gcc act04_q02.c -o act04_q02
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-05-SYS-PROG/Activity 04/program$ ./act04_q02
mysh >date
Sat Jan 25 09:13:43 PM +07 2025
mysh >cal 12 2021
    December 2021
Su Mo Tu We Th Fr Sa
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

mysh >exit
reisenx@reisenx-VirtualBox:~/Documents/GitHub/2110313-05-SYS-PROG/Activity 04/program$
```