운영체제 과제 4

※ magics 시스템을 사용할 때에 로그인 직후에 다음과 같이 프롬프트를 변경한 후 사용하며 동작 결과를 보여줄 때에 프롬프트도 함께 보여주시오.

```
$ PS1='\u:\W \!\$ '
```

1. **(IPC)** 뒤에 첨부한 프로그램을 분석하고 실행시켜보시오. 그리고 이 프로그램과 각종 자료들(과목 홈페이지 게시 자료, 인터넷 검색자료 포함)을 참고하여 POSIX의 shared memory와 message queue를 사용한 interprocess communication 기능을 비교 설명하시오.

2. **(pipe)** (1) Unix/Linux에서 교과서 3장의 Figure 3–25 프로그램을 작성하여 실행시키고, pipe 기능의 사용법에 대해서 알아보시오.
   (2) 이를 참고하고 연습문제(Programming Problem) 3.19 번의 pipe 기능을 이용한 응용 프로그램을 작성하시오.
   (**프로그램 내용**) 2개의 pipe를 만든 후에 child process를 생성한다. parent는 1st pipe로 string을 write하고 2nd pipe로부터 string을 read하여 이를 출력한다. child process는 1st pipe에서 sting을 read하고 읽은 string의 대소문자를 서로 반대로 바꾸어서 2nd pipe에 write한다.

3. **(간단한 shell 작성)** UNIX/Linux 운영체제 환경에서 명령어를 입력받아서 실행시키는 기능을 갖는 프로그램 mysh을 작성하시오. (힌트: mysh에서 fork와 execvp를 사용하여 입력받은 명령어를 실행시킨다. 명령어의 인수가 가변개수이므로 execlp보다 execvp를 사용하는 것이 편리하다, exec계열 시스템 호출 사용법은 [process] 참고자료 및 인터넷 자료를 참조하시오.) 이 shell은 exit 명령어 또는 Ctrl–D를 입력받으면 종료한다. (동작 확인을 할 때에 다양한 명령어를 입력해보시오.)

```
$ mysh
mysh〉 command arg1 ...          mysh〉은 프롬프트임, 명령어를 입력받음
command 실행                     입력 명령어를 실행함
mysh〉 ls –l
...
mysh〉 exit 또는 Ctrl–D           exit 또는 Ctrl–D가 입력되면 mysh을 종료함
$
```

〈참고 프로그램〉

```
--- UNIX/Linux Shared memory를 사용하는 프로그램 ---
/* Simple program demonstrating shared memory in POSIX systems. */
#include <stdio.h>
#include <sys/shm.h>
#include <sys/stat.h>

int main()
{
        int pid;                /* the identifier for the shared memory segment */
        int segment_id;         /* a pointer to the shared memory segment */
        char* shared_memory;    /* the size of the shared memory segment (byte) */
        const int segment_size = 4096;

        /** allocate  a shared memory segment */
        segment_id = shmget(IPC_PRIVATE, segment_size, S_IRUSR | S_IWUSR);
        printf("create shared memory : segment_id = %d\n", segment_id);
```

```
        /** attach the shared memory segment */
        shared_memory = (char *) shmat(segment_id, NULL, 0);

        pid = fork();
        if (pid < 0)
                return -1;
        else if (pid == 0) {  /* child */
                /** write a message to the shared memory segment   */
                sprintf(shared_memory, "Hello Parent");
                /** now detach the shared memory segment */
                if ( shmdt(shared_memory) == -1) {
                        fprintf(stderr, "Unable to detach\n");
                }
        } else {  /* parent */
                wait(NULL);
                /** now print out the string from shared memory */
                printf("*%s*\n", shared_memory);
                /** now detach the shared memory segment */
                if ( shmdt(shared_memory) == -1) {
                        fprintf(stderr, "Unable to detach\n");
                }

                /** now remove the shared memory segment */
                shmctl(segment_id, IPC_RMID, NULL);
        }

        return 0;
}
```

--- POSIX message queue를 사용하는 프로그램 (message send) ---

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MSGSZ    128

/* Declare the message structure. */
typedef struct msgbuf
{
        long mtype;
        char mtext[MSGSZ];
} message_buf;

int main(void)
{
        int msqid;
        int msgflg = IPC_CREAT | 0666;
        key_t key;
        message_buf sbuf;
        size_t buf_length;

        /* Get the message queue id for the "name" 1234,
         * which was created by the server.  */
        key = 1234;
        fprintf (stderr, "\nmsgget: Calling msgget(%#lx,%#o)\n", key, msgflg);
        if ((msqid = msgget (key, msgflg)) < 0) {
                perror ("msgget");
                exit (1);
        } else
                fprintf (stderr, "msgget: msgget succeeded: msqid = %d\n", msqid);

        /* We'll send message type 1 */
        sbuf.mtype = 1;
        strcpy (sbuf.mtext, "Did you get this?");
        buf_length = strlen (sbuf.mtext) + 1;
```

```
        /* Send a message. */
        if (msgsnd (msqid, &sbuf, buf_length, IPC_NOWAIT) < 0) {
                printf  ("%d,  %d,  %s,  %d\n",  msqid,  sbuf.mtype,  sbuf.mtext,
buf_length);
                perror ("msgsnd");
                exit (1);
        } else
                printf ("Message: \"%s\" Sent\n", sbuf.mtext);

        exit (0);
}
```

--- POSIX message queue를 사용하는 프로그램 (message receive) ---

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>

#define MSGSZ    128

/* Declare the message structure. */
typedef struct msgbuf
{
        long mtype;
        char mtext[MSGSZ];
} message_buf;

int main (void)
{
        int msqid;
        key_t key;
        message_buf rbuf;

        /* Get the message queue id for the "name" 1234,
         * which was created by the server. */
        key = 1234;

        if ((msqid = msgget (key, 0666)) < 0) {
                perror ("msgget");
                exit (1);
        }
        /* Receive an answer of message type 1. */
        if (msgrcv (msqid, &rbuf, MSGSZ, 1, 0) < 0) {
                perror ("msgrcv");
                exit (1);
        }
        /* Print the answer. */
        printf ("Received Message: %s\n", rbuf.mtext);
        exit (0);
}
```