Spring 2019 Operating Systems
Lab 9-2: Message Queue

0. 과제 수행 전 유의사항
프로그래밍은 자기 손으로 작성해야 한다. 아이디어를 서로 공유하고 도움을 받는 것은 좋으나, 다른 사람의 코드를 그대로 가져다 쓰는 것은 자신에게도 손해이고 다른 사람에게 피해를 끼치는 결과를 낳게되므로 허용하지 않는다. 과제 체크 시 두 사람의 코드가 유사하면, 두 사람 모두에게 과제 점수를 부여할 수 없으니 유의바란다.

1. 목표
메시지 큐에 대해서 실습한다.

2. 실습
다음 코드를 작성하고 테스트한다.

[lab10_1.c]

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

#define SIZE 1024

void signal_handler(int signo);

struct {
    long type;
    char data[SIZE];
} msg_data = {1, "Operating system concepts"};

int msqid;

main()
{
    if ((msqid = msgget((key_t)1234, IPC_CREAT|0666)) == -1) {
        perror("msgget failed");
        exit(1);
    }

    if (msgsnd(msqid, &msg_data, strlen(msg_data.data), 0) == -1) {
        perror("msgsnd failed");
        exit(1);
    }

    signal(SIGINT, signal_handler);
    pause();
}

void signal_handler(int signo)
{
    if (msgctl(msqid, IPC_RMID, 0) == -1) {
        perror("msgctl failed");
        exit(1);
    }
    wait(0);
}
```

[lab10_2.c]

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

#define SIZE 1024

struct {
    long type;
    char data[SIZE];
} msg_data;
```

```c
main()
{
    int msqid;
    struct msqid_ds msg_stat;

    if ((msqid = msgget((key_t)1234, IPC_CREAT|0666)) == -1) {
        perror("msgget failed");
        exit(1);
    }

    if (msgrcv(msqid, &msg_data, SIZE, 0, 0) == -1) {
        perror("msgrcv failed");
        exit(1);
    }
    printf("data read from message queue: %s\n", msg_data.data);

    if (msgctl(msqid, IPC_STAT, &msg_stat) == -1) {
        perror("msgctl failed");
        exit(1);
    }

    kill(msg_stat.msg_lspid, SIGINT);

    exit(0);
}
```

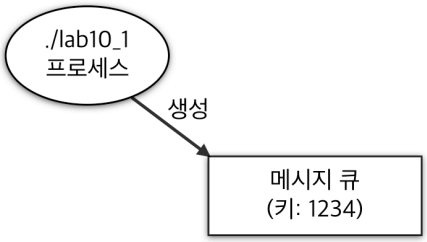
주의사항: msgget( ) 함수에 쓰이는 (key_t)1234 이 부분의 키 값은 시스템 전체에서 공유된다. 따라서, 여러 사람이 사용하는 서버에서는 다른 사람이 이 키 값을 이용하여 데이터를 가져갈 수 있으니 자신만 쓸 수 있는 키 값으로 바꾼다. ex) 20114231

- 컴파일
$ gcc lab10_1.c -o lab10_1
$ gcc lab10_2.c -o lab10_2

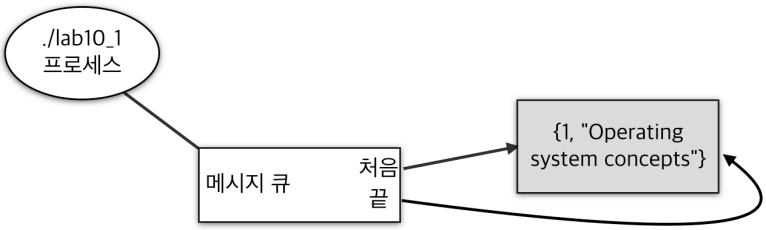
- 실행
$ ./lab10_1 &
$ ./lab10_2

- 위 프로그램의 동작 과정은 다음과 같다.

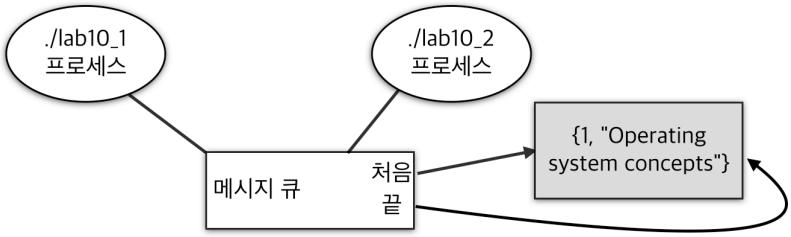
(1) ./lab10_1 의 프로세스는 msgget( ) 으로 키가 1234인 메시지 큐를 생성한다.



(2) msg_data 의 메시지를 msgsnd( ) 로 메시지 큐에 전송해 메시지가 연결되고 ./lab10_1 의 프로세스는 pause( ) 에 의해 시그널을 기다린다.



(3) ./lab10_2 의 프로세스는 msgget( ) 으로 키가 1234인 메시지 큐에 접근한다.



키가 1234인 메시지 큐는 생성되어 있으므로 아래 코드를

```
msqid = msgget((key_t)1234, IPC_CREAT|0666);
```

아래와 같이 바꿔도 동작한다.

```
msqid = msgget((key_t)1234, 0);
```

(4) ./lab10_2 의 프로세스는 msgrcv( ) 로 메시지를 읽어 msg_data 에 저장한다. 이 때, 읽혀진 메시지는 메시지 큐에서 삭제된다.

그리고 msgctl( ) 로 메시지 큐의 정보를 얻어와 msg_stat 에 저장한다.

(5) msg_stat.msg_lspid 는, 마지막으로 msgsnd( ) 함수를 호출한 프로세스의 프로세스 ID를 의미하므로 ./lab10_1 의 프로세스 ID가 된다. kill( ) 로 ./lab10_1 의 프로세스에 SIGINT 시그널을 보내고 ./lab10_2 의 프로세스는 종료된다.

3. 과제

다음 코드는 메시지 큐에 우선순위가 있는 메시지를 전달하는 코드이다. 이 코드를 동작해보고 과정을 설명하시오.

[lab10_3.c]

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

#define SIZE 1024

struct {
    long type;
    char data[SIZE];
} msg_data;

main()
{
    int msqid, data_len;
    char buffer[SIZE];

    if ((msqid = msgget((key_t)1234, IPC_CREAT|0666)) == -1) {
        perror("msgget failed");
        exit(1);
    }

    while (!0) {
        printf("input data => ");
        scanf(" %[^\n]", buffer);

        if (!strcmp(buffer, "quit")) {
            break;
        }

        printf("input priority => ");
        scanf("%ld", &(msg_data.type));
        strcpy(msg_data.data, buffer);
        if ((msgsnd(msqid, &msg_data, strlen(msg_data.data), 0)) == -1) {
            perror("msgsnd failed");
            exit(1);
        }
    }

    exit(0);
}
```

[lab10_4.c]

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

#define SIZE 1024
#define PRIOR 10

struct {
    long type;
    char data[SIZE];
} msg_data;

main()
{
    int msqid, data_len;

    if ((msqid = msgget((key_t)1234, IPC_CREAT|0666)) == -1) {
        perror("msgget failed");
        exit(1);
    }

    while (!0) {
        if ((data_len = msgrcv(msqid, &msg_data, SIZE, (-1*PRIOR), IPC_NOWAIT)) == -1) {
            if (errno == ENOMSG) {
                printf("no more massages\n");
                break;
            }
            perror("msgrcv failed");
            break;
        }

        msg_data.data[data_len] = '\0';
        printf("data: %s [%ld]\n", msg_data.data, msg_data.type);
    }

    if (msgctl(msqid, IPC_RMID, 0) == -1) {
        perror("msgctl failed");
        exit(1);
    }

    exit(0);
}
```

주의사항: msgget( ) 함수에 쓰이는 (key_t)1234 이 부분의 키 값은 시스템 전체에서 공유된다. 따라서, 여러 사람이 사용하는 서버에서는 다른 사람이 이 키 값을 이용하여 데이터를 가져갈 수 있으니 자신만 쓸 수 있는 키 값으로 바꾼다. ex) 20114231

- 컴파일

```
$ gcc lab10_3.c -o lab10_3
$ gcc lab10_4.c -o lab10_4
```

- 실행 예

```
s_guest@A1409-OSLAB-01:~/lab10$ ./lab10_3
input data => bye bye !!
input priority => 10
input data => I'm your message queue
input priority => 1
input data => 4 dollars !!!!
input priority => 8
input data => 4 dollars....
input priority => 3
input data => ok 4 dollar thank you !!
input priority => 9
input data => quit
s_guest@A1409-OSLAB-01:~/lab10$ ./lab10_4
data: I'm your message queue [1]
data: 4 dollars.... [3]
data: 4 dollars !!!! [8]
data: ok 4 dollar thank you !! [9]
data: bye bye !! [10]
no more messages
```

4. 제출

자신이 정리한 코드에 주석을 붙여서 제출한다.