Spring 2019 Operating Systems
Lab 10: Shared Memory

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
공유 메모리에 대하여 실습한다.

2. 실습
다음 코드를 작성하고 테스트한다.

[lab11_1.c]

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

#define SIZE 1024

void signal_handler(int signo);
int shmid;

main()
{
    void *shmaddr;

    if ((shmid = shmget((key_t)1234, SIZE, IPC_CREAT|0666)) == -1) {
        perror("shmid failed");
        exit(1);
    }

    if ((shmaddr = shmat(shmid, (void *)0, 0)) == (void *)-1) {
        perror("shmat failed");
        exit(1);
    }

    strcpy((char *)shmaddr, "Operating system concepts");

    if (shmdt(shmaddr) == -1) {
        perror("shmdt failed");
        exit(1);
    }

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

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

[lab11_2.c]

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

#define SIZE 1024

main()
{
    int shmid;
    void *shmaddr;
    struct shmid_ds shm_stat;

    if ((shmid = shmget((key_t)1234, SIZE, IPC_CREAT|0666)) == -1) {
        perror("shmid failed");
        exit(1);
    }

    if ((shmaddr = shmat(shmid, (void *)0, 0)) == (void *)-1) {
        perror("shmat failed");
        exit(1);
    }

    printf("data read from shared memory: %s\n", (char *)shmaddr);

    if (shmctl(shmid, IPC_STAT, &shm_stat) == -1) {
        perror("shmctl failed");
        exit(1);
    }

    if (shmdt(shmaddr) == -1) {
        perror("shmdt failed");
        exit(1);
    }

    kill(shm_stat.shm_cpid, SIGINT);

    exit(0);
}
```

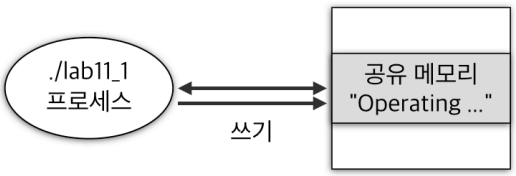
주의사항: (key_t)1234 이 부분의 키 값은 시스템 전체에서 공유된다. 따라서, 여러 사람이 사용하는 서버에서는 다른 사람이 이 키 값을 이용하여 데이터를 가져갈 수 있으니 자신만 쓸 수 있는 키 값으로 바꾼다. ex) 20114231

- 컴파일
$ gcc lab11_1.c -o lab11_1
$ gcc lab11_2.c -o lab11_2

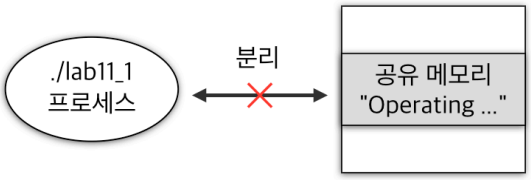
- 실행
$ ./lab11_1 &
$ ./lab11_2

- 위 프로그램의 동작 과정은 다음과 같다.

(1) ./lab11_1 의 프로세스는 shmget( ) 함수에 의해 1234 키를 갖는 공유 메모리를 생성하고 식별자를 반환받는다. shmat( ) 함수에 의해 자신의 메모리 영역에 공유 메모리를 첨부(설정)하고 공유 메모리(주소 shmaddr)에 "Operating system concepts" 를 쓴다.



(2) shmdt( ) 에 의해 ./lab11_1 프로세스의 자신의 메모리 영역에 첨부(설정)되어있던 공유 메모리를 분리한다.



그리고 ./lab11_1 프로세스는 pause( ) 함수 호출에 의해 시그널을 기다린다.

(3) ./lab10_1 프로세스는 shmget( ) 함수에 의해 1234 키의 공유 메모리에 대한 식별자를 반환받고, shmdt( ) 에 의해 자신의 메모리 영역에 공유 메모리를 첨부(설정)한다.

(4) 공유 메모리에 저장된 데이터를 읽어 출력한다.

(5) shmctl( ) 에 의해 공유 메모리에 대한 정보를 가져오고, 사용이 끝났으므로 shmdt( ) 에 의해 공유 메모리를 분리한다.

(6) ./lab11_2 프로세스는 kill( ) 함수에 의해 shm_stat.shm_cpid 에게 SIGINT 시그널을 보내고 종료하는데, shm_stat.shm_cpid 는 공유 메모리를 생성한 프로세스의 프로세스 ID 이므로 ./lab11_1 프로세스를 의미한다.

(7) SIGINT 시그널을 받은 ./lab11_1 프로세스는 pause( ) 함수에서 대기하고 있다가 signal 을 받고 깨어난 후, 설정된 SIGINT 시그널에 설정된 signal_handler( ) 함수를 호출한다. 이후, shmctl( ) 함수에 의해 공유 메모리를 제거한다.

3. 과제

다음 코드는 공유 메모리를 이용하여 여러 프로세스에서 보내는 데이터를 받는 예제이다. 지금까지 살펴본 예제에서는 두 개의 프로세스간에 통신을 했는데, 이번에는 두 개 이상의 프로세스에서 공유 메모리에 쓰는 데이터를 다른 한 프로세스가 읽어 출력하는 예제를 살펴본다.

[lab11_3.c]

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

#define SIZE 1024

main()
{
    struct check_data {
        int check;
        char data[SIZE];
    } *shared_data;
    int shmid;
    void *shmaddr;
    struct shmid_ds shm_stat;

    if ((shmid = shmget((key_t)1234, sizeof(struct check_data), IPC_CREAT|0666)) == -1) {
        perror("shmid failed");
        exit(1);
    }

    if ((shmaddr = shmat(shmid, (void *)0, 0)) == (void *)-1) {
        perror("shmat failed");
        exit(1);
    }

    shared_data = (struct check_data *)shmaddr;
    shared_data->check = 1;

    while (!0) {
        if (shared_data->check) {
            printf("data read from shared memory: %s\n", shared_data->data);
            sleep(1);
            shared_data->check = 0;
            if (!strncmp(shared_data->data, "quit", 4)) {
                if (shmctl(shmid, IPC_STAT, &shm_stat) == -1) {
                    perror("shmctl failed");
                    exit(1);
                }
                if (shm_stat.shm_nattch == 1) {
                    break;
                }
            }
        }
        sleep(1);
    }

    if (shmdt(shmaddr) == -1) {
        perror("shmdt failed");
        exit(1);
    }

    if (shmctl(shmid, IPC_RMID, 0) == -1) {
        perror("shmctl failed");
        exit(1);
    }
    exit(0);
}
```

[lab11_4.c]

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

#define SIZE 1024

main()
{
    struct check_data {
        int check;
        char data[SIZE];
    } *shared_data;
    int shmid;
    void *shmaddr;
    char buffer[SIZE];

    if ((shmid = shmget((key_t)1234, SIZE, IPC_CREAT|0666)) == -1) {
        perror("shmid failed");
        exit(1);
    }

    if ((shmaddr = shmat(shmid, (void *)0, 0)) == (void *)-1) {
        perror("shmat failed");
        exit(1);
    }

    shared_data = (struct check_data *)shmaddr;

    while (!0) {
        while (shared_data->check) {
            sleep(1);
            printf("waiting....\n");
        }
        printf("input data ==> ");
        fgets(buffer, SIZE, stdin);

        strncpy(shared_data->data, buffer, SIZE);
        shared_data->check = 1;
        if (!strncmp(shared_data->data, "quit", 4)) {
            break;
        }
    }

    if (shmdt(shmaddr) == -1) {
        perror("shmdt failed");
        exit(1);
    }
    exit(0);
}
```

주의사항: 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 lab11_3.c -o lab11_3
$ gcc lab11_4.c -o lab11_4
```

- 실행 예

```
s_guest@A1409-OSLAB-01:~/lab11$ ./26_7 &
[1] 19534
s_guest@A1409-OSLAB-01:~/lab11$ data read from shared memory:

s_guest@A1409-OSLAB-01:~/lab11$ ./26_8
input data ==> Hello?
data read from shared memory: Hello?

waiting....
waiting....
input data ==> I'm shared memory
data read from shared memory: I'm shared memory

waiting....
waiting....
input data ==> dog honey
data read from shared memory: dog honey

waiting....
waiting....
input data ==> quit
s_guest@A1409-OSLAB-01:~/lab11$ data read from shared memory: quit


[1]+  Done                    ./26_7
```

4. 제출

자신이 정리한 코드에 주석을 붙여서 제출한다.