과목명	운영체제	분반	X	담당교수	김화성 교수님
학과	전자통신공학과	학번	2016707079	이름	하상천
과제명: H/W 2 - 프로세스2 (POSIX SM API)					

1. 과제설명 (사용자 요구사항 기술: 과제에 대한 설명 및 목표)

3장 강의 자료에서 기술된 Producer-Consumer Problem (Shared-Memory Solution)을 POSIX SM API를 이용하여 구현하고 테스트하시요.

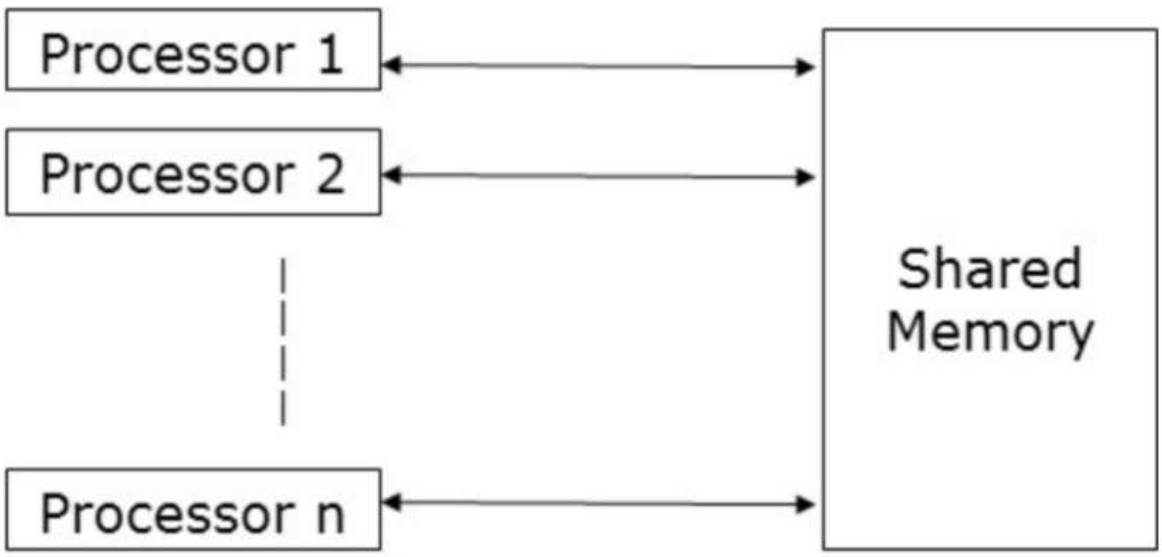
단,

1) 구현 언어는 C언어를 사용함.

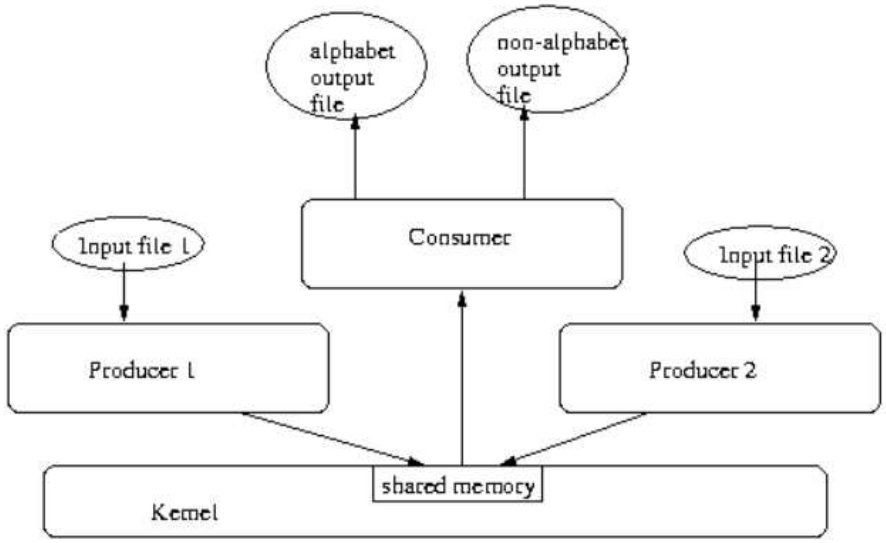
2) 과제물은 제출 형식 파일을 이용하여 업로드하고, 파일 이름은 과제2.hwp로 하여 업로드함

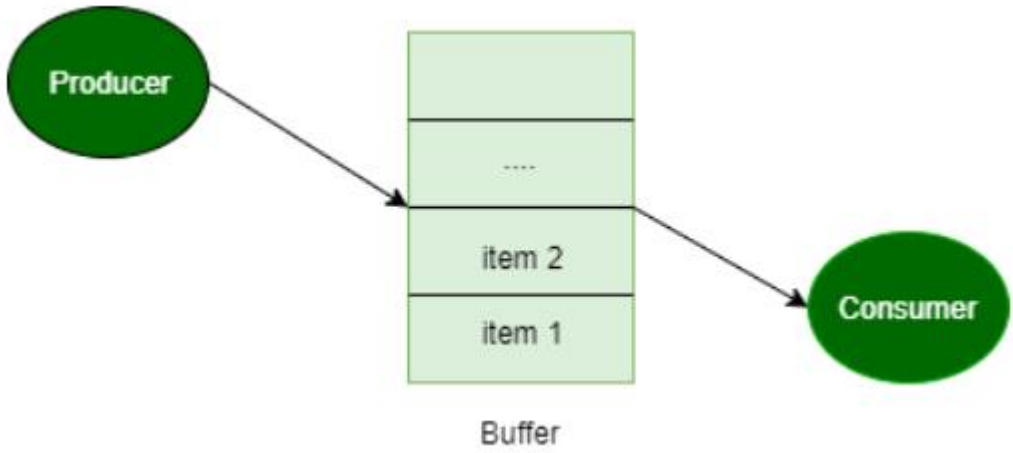
3) 소스 코드 파일은 과제2.c로 하여 업로드 함.

2. 사용자 요구사항을 정형적 방법으로 기술 (UML, Pseudo code, 그림등을 이용하여 기술)



3. 알고리즘 및 자료구조 설계 내용





4. 소스코드 설명 (직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명)

```
shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
ftruncate(shm_fd, SIZE);
```

shm_open을 통해서 POSIX shared memory object를 open한다. 두 번째 파라미터인 O_CREAT | O_RDWR를 통해서 존재하지 않으면 생성하고, 존재하면 read-write access용으로 open 한다. ftruncate를 통해서 shm_fd의 크기를 조절해준다.

```
ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
sprintf(ptr, "%s", message0);
ptr += strlen(message0);
sprintf(ptr, "%s", message1);
ptr += strlen(message1);
sprintf(ptr, "%s", message2);
ptr += strlen(message2);
```

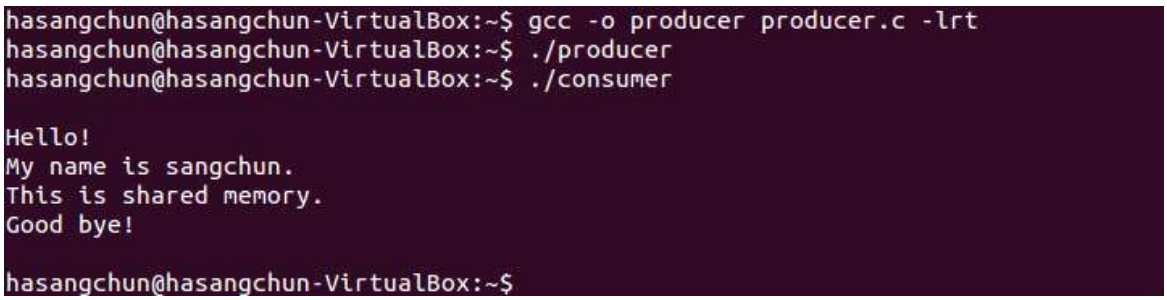
mmap을 통해 해당되는 메모리에 맵핑하도록 커널에 요청한다. 세 번째 파라미터는 맵핑에 원하는 메모리 보호 정책인데 PROT_WRITE로 쓰기 가능하도록 하였다. 네 번째 파라미터는 맵핑 유형과 동작 구

성요소인데 MAP_SHARED로 하여 동일 파일을 맵핑한 모든 프로세스들이 공유할 수 있도록 하였다. sprintf를 통해 메시지를 ptr로 복사한다. strlen함수는 문자열의 길이를 구하는 함수이다. ptr += strlen(message0)을 해주는 이유는 mmap 함수의 리턴값이 맵핑이 시작되는 실제 메모리 주소 값이기 때문이다.

```
shm_fd = shm_open(name, O_RDONLY, 0666);
ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
printf("%s", (char*)ptr);
```

consumer에서는 shm_open을 O_RDONLY로 하여 read access용으로 open한다. mmap함수에서도 MAP_SHARED는 동일하지만, 세 번째 파라미터는 PROT_READ로 하여 읽기 가능하도록 하였다. ptr로 읽어본 값을 char 포인터형으로 형변환해주고 출력해준다.

5. 실행결과 및 설명 (실행 결과를 캡쳐하여 첨부한 후 설명)



(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

작성된 프로그램을 실행한 결과 위와 같은 화면이 출력 되었다.

6. 고찰 (과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점)

이번 과제는 수업시간에 배운 Producer-Consumer Problem (Shared-Memory Solution)을 POSIX SM API를 이용하여 구현하고 테스트하는 것이었다. shared memory API를 처음 사용해봐서 조금 낯설었지만, 강의자료를 통해 함수의 역할들을 공부하고 인터넷과 책을 찾아보니 익숙해졌다. const를 이용해서 상수로 지정하였는데 처음에는 #define을 이용했었다. 둘의 차이를 몰라 찾아본 결과, define은 메모리에 올라가지 않고 치환해놓은 개념이라면, const는 type을 지정해줄 수 있어서 에러를 확인하기 쉽고 메모리가 할당되었다. 또한 const는 일반 변수와 생성 및 동작 메커니즘이 같았다. 서로 장단점이 있지만 이번 코드를 작성할 때는 const를 이용해서 상수를 지정해주었다. 왜냐하면 type을 지정하여 error를 쉽게 확인할 수 있기 때문이다. shm_open, ftruncate, mmap, shm_unlink 등 여러 함수들을 처음 사용해봐서 어색했지만, 강의자료에 설명이 잘 되어있고 인터넷에 파라미터 정보와 리턴값이 잘 나와 있어서 확인하면서 코딩하였다. 수업시간에 배운 IPC(interprocess communication)의 두 가지 모델 중 한 가지 모델인 shared memory 방법을 직접 C언어로 구현해보니 수업내용이 더 잘 이해되는 것 같다.

7. 전체 소스코드 (글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성)

Producer 코드

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

```
#include<sys/shm.h>
#include<sys/stat.h>
#include<sys/mman.h>

void main() {
const int SIZE = 4096;  //Constant
const char *name = "shared_memory"
const char *message0 = "₩nHello₩n"
const char *message1 = "My name is sangchun.₩nThis is shared memory.₩n"
const char *message2 = "Good bye!₩n₩n"

int shm_fd;
void *ptr;

shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
ftruncate(shm_fd, SIZE);
ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
sprintf(ptr, "%s", message0);
ptr += strlen(message0);
sprintf(ptr, "%s", message1);
ptr += strlen(message1);
sprintf(ptr, "%s", message2);
ptr += strlen(message2);
}
```

Consumer 코드

```
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<sys/stat.h>
#include<sys/mman.h>

void main() {
const int SIZE 4096; //Constant
const char *name = "shared_memory"
int shm_fd;
void *ptr;
```

```
shm_fd = shm_open(name, O_RDONLY, 0666);
ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
printf("%s", (char*)ptr);
shm_unlink(name);
}
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)