세마포를 이용하여 Lock과 Condition Variable을 만들어 구성된 Mesa Style Monitor로 Reader & Writer Problem을 구현하시오.

**- Lock 만들기**

/\*hw3.h : lock function 선언 & 구현\*/

#define key\_m 2012343000 // 서버에서 작업할 때는 자기 학번 등을 이용하여 다른 사람의 키와 중복되지 않게 해야 함

#define key\_okR key\_m+1

#define key\_okW key\_m+2

void Acquire(int mutex){

p(mutex);

}

void Release(int mutex){

v(mutex);

}

/\* hw3R.c / hw3W.c lock 사용 예\*/

if ((lock = initsem(key\_m,1)) < 0) //초기값이 1(FREE)인 semaphore로 lock 만듦

exit(1);

Acquire(lock);

Release(lock);

**- Condition Variable 만들기**

/\*hw3.h : condition Variable function 선언 & 구현\*/

#define key\_m 2012343000 // 서버에서 작업할 때는 자기 학번 등을 이용하여 다른 사람의 키와 중복되지 않게 해야 함

#define key\_okR key\_m+1

#define key\_okW key\_m+2

void wait(int semid, int lock){

v(lock);

p(semid);

p(lock);

}

void signal(int semid){

v(semid);

}

void broadcast(int semid, int num){

int i;

for(i=0; i<num; i++){

v(semid);

}

}

/\* hw3R.c / hw3W.c condition Variable 사용 예\*/

if ((reader = initsem(key\_okR,1)) < 0) // reader semaphore 연결.(없으면 새로 만들어 연결. 초기값 1)

exit(1);

if ((writer = initsem(key\_okW,1)) < 0) // writer semaphore 연결.(없으면 새로 만들어 연결. 초기값 1)

exit(1);

**- Reader, Writer가 협력하며 관리하는 4개의 카운터(AR,AW,WR,WW)는 4개의 화일에 기록하기로 한다.)**

: ar(int num), aw(int num), wr(int num), ww(int num) function은 동일한 알고리즘을 사용해 만들었다.

각각 Parameter int num이 0이면 현재 ar/aw/wr/ww 개수를 return 하고

그렇지 않으면(num!=0 이면) 현재 ar/aw/wr/ww 개수에서 num만큼을 더해주고 ar/aw/wr/ww에 카운터 레코드를 출력한다.

레코드 출력 format : <변한ar/aw/wr/ww수>, <pid>, <실행된 순서>, <시간:Www Mmm dd hh:mm:ss yyyy>

1. AW & WW

/\*hw3.h : writer function 선언 & 구현\*/

int ActvieWriter(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("AW.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

fclose(fstream);

return val;

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

int WaitingWriter(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("WW.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

2. AR & WR

/\*hw3.h : reader function 선언 & 구현\*/

int ActiveReader(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("AR.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

int WaitingReader(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("WR.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

**- reader, writer 구현**

Header & C file download -> click 

1. writer – hw3R.c

#include "hw3.h"

void main(int argc, char \*\*argv){

int i;

int lock, reader, writer;

sleep(atoi(argv[1]));

if ((lock = initsem(key\_m,1)) < 0) // lock semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

if ((reader = initsem(key\_okR,1)) < 0) // reader semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

if ((writer = initsem(key\_okW,1)) < 0) // writer semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

Acquire(lock); // Lock Acquire()

while( (ActvieWriter(0) + ActiveReader(0)) > 0 ){

WaitingWriter(1);

wait(writer, lock);

WaitingWriter(-1);

}

ActvieWriter(1); //increment

Release(lock);

//Perform actual read/write access AccessDatabase(ReadWrite);

sleep(atoi(argv[2]));

Acquire(lock);

ActvieWriter(-1); //decrement

if( WaitingWriter(0) >0 )

signal(writer);

else if(WaitingReader(0)>0) {

broadcast(reader, WaitingReader(0));

}

Release(lock);

}

2. reader – hw3R.c

#include "hw3.h"

void main(int argc, char \*\*argv){

int i;

int lock, reader, writer;

sleep(atoi(argv[1]));

if ((lock = initsem(key\_m,1)) < 0) // lock semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

if ((reader = initsem(key\_okR,1)) < 0) // reader semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

if ((writer = initsem(key\_okW,1)) < 0) // writer semaphore 연결.(없으면 초기값을 1로 주면서 새로 만들어서 연결한다.)

exit(1);

Acquire(lock); // Lock Acquire()

while( (ActvieWriter(0) + WaitingWriter(0)) > 0 ){

WaitingReader(1);

wait(reader, lock);

WaitingReader(-1);

}

ActiveReader(1); //increment

Release(lock);

//Perform actual read-only access AccessDatabase(ReadOnly);

sleep(atoi(argv[2]));

Acquire(lock);

ActiveReader(-1); //decrement

if( ActiveReader(0)==0 && WaitingWriter(0)>0 )

signal(writer);

Release(lock);

}

3. header 전문 – hw3.h

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <errno.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define SEMPERM 0600

#define TRUE 1

#define FALSE 0

#define key\_m 2012343000

#define key\_okR key\_m+1

#define key\_okW key\_m+2

int count(){

int result;

FILE \*fstream = fopen("count.txt", "a+");

while(!feof(fstream))

fscanf(fstream, "%d", &result);

fprintf(fstream, "%d\n", result+1);

fclose(fstream);

return result;

}

//semaphore object

typedef union semun{

int val;

struct semid\_ds \*buf; //sem\_id : semaphore set associated with a semaphore ID. There is one semaphore set per semaphore ID.

ushort \*array;

} semun;

int initsem (key\_t semkey, int n) {

int status = 0, semid;

if ( (semid = semget (semkey, 1, SEMPERM | IPC\_CREAT | IPC\_EXCL)) == -1) { //semget : get a semaphore set identifier

if (errno == EEXIST) //errno : number of last error

semid = semget (semkey, 1, 0);

}

else{

semun arg;

arg.val = n;

status = semctl(semid, 0, SETVAL, arg); //semctl : control operation

}

if (semid == -1 || status == -1) {

perror("initsem failed");

return (-1);

}

return (semid);

}

int p (int semid) {

struct sembuf p\_buf;

p\_buf.sem\_num = 0;

p\_buf.sem\_op = -1;

p\_buf.sem\_flg = SEM\_UNDO;

if (semop(semid, &p\_buf, 1) == -1) {

perror ("p(semid) failed");

exit(1);

}

return (0);

}

int v (int semid)

{

struct sembuf v\_buf;

v\_buf.sem\_num = 0;

v\_buf.sem\_op = 1;

v\_buf.sem\_flg = SEM\_UNDO;

if (semop(semid, &v\_buf, 1) == -1)

{

perror ("v(semid) failed");

exit(1);

}

return (0);

}

void Acquire(int mutex){

p(mutex);

}

void Release(int mutex){

v(mutex);

}

void wait(int semid, int lock){

v(lock);

p(semid);

p(lock);

}

void signal(int semid){

v(semid);

}

void broadcast(int semid, int num){

int i;

for(i=0; i<num; i++){

v(semid);

}

}

int ActvieWriter(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("AW.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

fclose(fstream);

return val;

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

int WaitingWriter(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("WW.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

int ActiveReader(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("AR.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

}

}

int WaitingReader(int num){

char str[50];

int val;

time\_t t;

FILE \*fstream = fopen("WR.txt", "a+");

if(fstream == NULL)

exit(1);

while(!feof(fstream))

fgets(str, 50, fstream);

val=atoi(str);

if(num==0){

return val;

fclose(fstream);

}

else{

time(&t);

fprintf(fstream, "%d %d %d %s", val+num, getpid(), count(), ctime(&t));

fclose(fstream);

return 0;

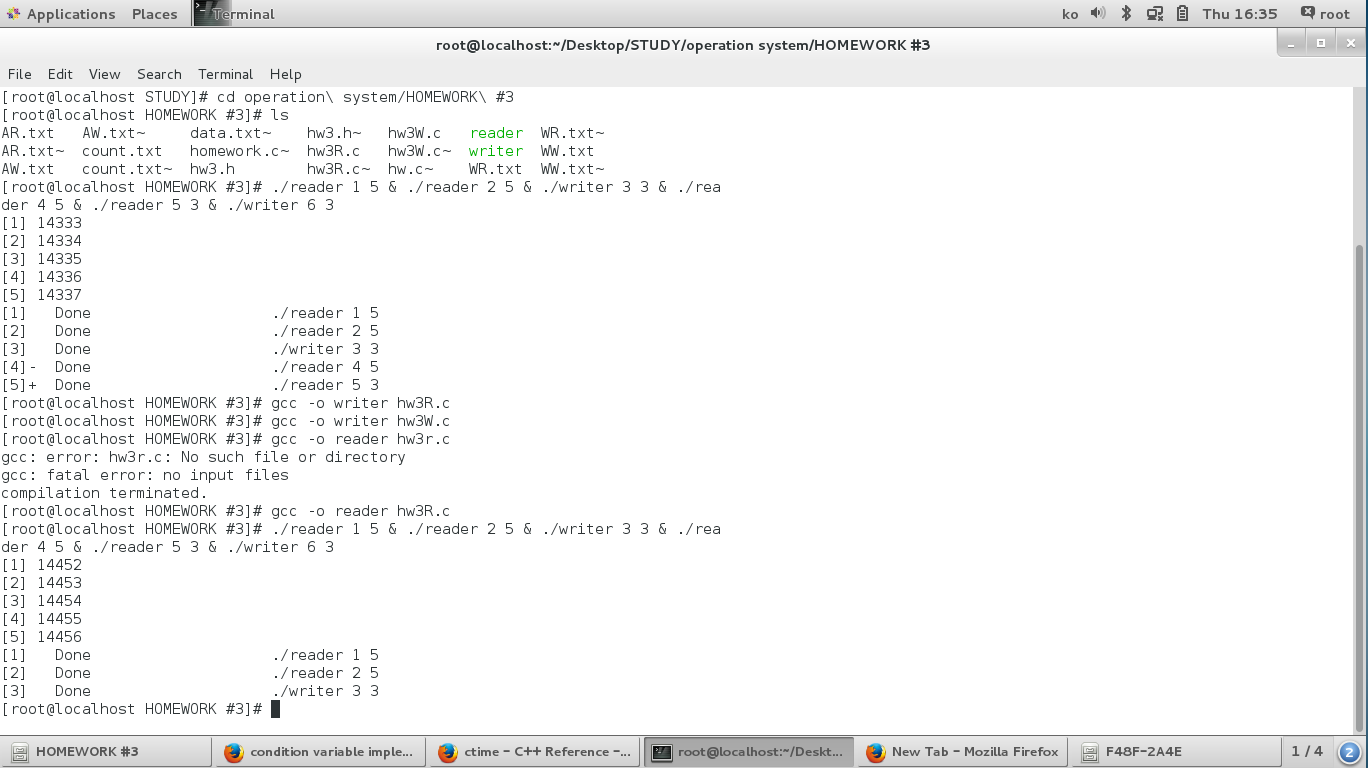
}

}

**- 실행 및 분석**

위와 같이 실행할 때 전체 실행시간이 몇 초이며 누가 가장 늦게 끝나는가를 실제 실행된 기록으로 설명하시오.

1. 실행 스크린샷



결과 다운받으려면 밑의 파일을 클릭하세요\_



2. 실행 표

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 실행순서 | PID | 시간 | AR | AW | WR | WW | CODE |
| 0 | 14452 | 16:17:41 | 1 |  |  |  | ActiveReader(1); |
| 1 | 14453 | 16:17:42 | 2 |  |  |  | ActiveReader(1); |
| 2 | 14454 | 16:17:43 |  |  |  | 1 | while( (ActvieWriter(0) + ActiveReader(0)) > 0 ){  WaitingWriter(1);  wait(writer, lock);  WaitingWriter(-1);  } |
| 3 | 16:17:43 |  |  |  | 0 |
| 4 | 16:17:43 |  |  |  | 1 |
| 5 | 14455 | 16:17:44 |  |  | 1 |  | while( (ActvieWriter(0) + WaitingWriter(0)) > 0 ){  WaitingReader(1);  wait(reader, lock);  WaitingReader(-1);  } |
| 6 | 16:17:44 |  |  | 0 |  |
| 7 | 16:17:44 |  |  | 1 |  |
| 8 | 14456 | 16:17:45 |  |  | 2 |  | while( (ActvieWriter(0) + WaitingWriter(0)) > 0 ){  WaitingReader(1);  wait(reader, lock); |
| 9 | 14452 | 16:17:46 | 1 |  |  |  | ActiveReader(-1); |
| 10 | 14457 | 16:17:46 |  |  |  | 2 | while( (ActvieWriter(0) + ActiveReader(0)) > 0 ){  WaitingWriter(1);  wait(writer, lock); |
| 11 | 14453 | 16:17:47 | 0 |  |  |  | ActiveReader(-1);  if( ActiveReader(0)==0 && WaitingWriter(0)>0 )  signal(writer); |
| 12 | 14454 | 16:17:47 |  |  |  | 1 | WaitingWriter(-1); ActvieWriter(1); ActvieWriter(-1); if( WaitingWriter(0) >0 )  signal(writer); |
| 13 | 16:17:47 |  | 1 |  |  |
| 14 | 16:17:50 |  | 0 |  |  |
| 15 | 14457 | 16:17:50 |  |  |  | 0 | WaitingWriter(-1); ActvieWriter(1); ActvieWriter(-1); else if(WaitingReader(0)>0) {  broadcast(reader, WaitingReader(0)); } |
| 16 | 16:17:50 |  | 1 |  |  |
| 17 | 16:17:53 |  | 0 |  |  |
| 18 | 14455 | 16:17:53 |  |  | 1 |  | WaitingReader(-1); ActiveReader(1); |
| 19 | 16:17:53 | 1 |  |  |  |
| 20 | 14456 | 16:17:53 |  |  | 0 |  | WaitingReader(-1); ActiveReader(1); ActiveReader(-1); |
| 21 | 16:17:53 | 2 |  |  |  |
| 22 | 16:17:56 | 1 |  |  |  |
| 23 | 14455 | 16:17:58 | 0 |  |  |  | ActiveReader(-1); |