|  |
| --- |
| **국민대학교 컴퓨터공학부** |
| Operating System |
| **Homework #3 Synchronization by Semaphore** |

|  |
| --- |
| 20093267 김 성 근  2014-04-14 |

**1. Solve Race Condition Problem [ Using Java – Failure ]**

첫 번째 과제 Race Condition 을 만들기 위해서 Java를 이용하였습니다. 따라서 이 Condition을 해결하기 위하여 Java의 Semaphore를 찾아보았고 이것을 활용해서 구현해 보았습니다.

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.util.concurrent.Semaphore;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** RaceCondition {

**private** **static** Logger *logger* = LoggerFactory.*getLogger*(RaceCondition.**class**);

**private** **final** Semaphore semaphore;

**public** RaceCondition(**int** count) {

**this**.semaphore = **new** Semaphore(count);

}

**public** **void** improveFileWrite(**int** value) **throws** InterruptedException {

semaphore.acquire();

**try** {

fileWriter(value);

} **finally** {

semaphore.release();

}

}

**public** **void** improveFileRead() **throws** InterruptedException {

semaphore.acquire();

**try** {

fileReader();

} **finally** {

semaphore.release();

}

}

**public** **void** fileWriter(**int** value) {

BufferedWriter file = **null**;

FileWriter writer = **null**;

**try** {

*logger*.info("Write : " + value);

writer = **new** FileWriter("input.txt", **true**);

file = **new** BufferedWriter(writer);

file.write("\n" + value);

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

file.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

writer.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

Code 1 RaceCondition.java

**public** String fileReader() {

BufferedReader file = **null**;

FileReader reader = **null**;

**try** {

reader = **new** FileReader("input.txt");

file = **new** BufferedReader(reader);

String line = **null**;

String last = **null**;

**while** ((line = file.readLine()) != **null**) {

last = line;

}

**return** last;

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

file.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

reader.close();

} **catch** (IOException e) {

e.printStackTrace();

}

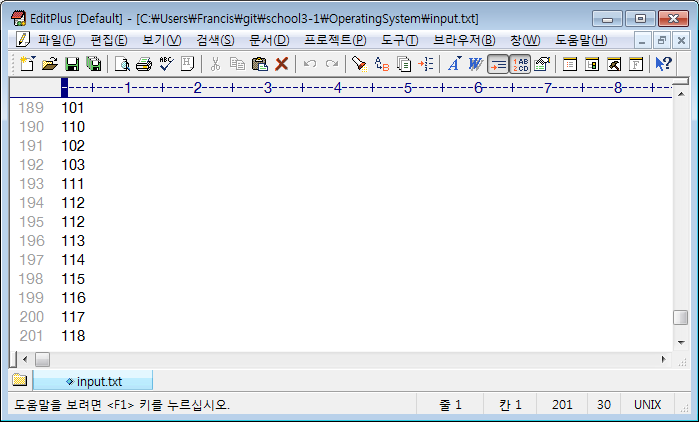
}

**return** **null**;

}

}

Code 2 RaceCondition.java cont.



Screen 1 결과 화면 (Using Java)

위와 같이 구현을 해보고 실행을 한 후에 여기서의 Semaphore의 구현은 Race Condition의 문제를 해결하기 위한 것이 아니라는 것을 알았습니다. 현재 Java 에서의 Semaphore는 Program 안에서의 Semaphore 공유였고, 이것은 우리가 하려고 하는 프로그램 간의 Semaphore 공유는 아니라는 것을 알게 되었습니다. Race Condition의 문제를 해결하기 위해서는 두 프로그램이 서로 돌면서 Semaphore를 공유해야 하는데 이것을 Java로 구현 할 수 있을지 여부를 찾아 보았지만, 해결방법을 찾을 수 없었습니다.

**2. Solve Race Condition Problem [ Using C – Success ]**

Java로 Semaphore 구현을 실패 했기 때문에 C로 RaceCondition 을 다시 작성하였습니다. 그리고 제공해주신 Advance IPC를 이용하기 위해 Linux 환경에서 해당 작업을 수행하였습니다. 먼저 Linux에서 사용하기 위해 작성된 C Code입니다.

#include "RaceCondition.h"

void main(){

addFile();

return;

}

Code 3 main.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define FILE\_NAME "input.txt"

void addFile();

int readFile(FILE \*file);

void writeFile(FILE \*file, int value);

Code 4 RaceCondition.h

#include "RaceCondition.h"

void addFile(){

FILE \*readf = NULL, \*writef = NULL;

int i;

printf("Race Condition Start\n");

for(i=0; i< 100; i++){

int value;

value = readFile(readf);

writeFile(writef, value);

}

printf("Race Condition End\n");

}

int readFile(FILE \*file){

int value;

if( (file = fopen(FILE\_NAME, "r")) == NULL){

return 0;

}

while(!feof(file)){

fscanf(file, "%d", &value);

}

fclose(file);

return value;

}

void writeFile(FILE \*file, int value){

if( (file = fopen(FILE\_NAME, "a")) == NULL){

return;

}

fprintf(file, "\n%d", value+1);

fclose(file);

}

Code 5 RaceCondition.c

이후 Semaphore 사용에 있어서 Critical Section 부분을 간략하기 위하여 함수 형태로 불러올 수 있게 작성하였습니다.

해당 코드를 실행하기 위하여 **Code 6** 과 같이 Linux Shell을 이용하여 동시에 3개를 실행하였고, 실행 결과는 **Screen 2** 와 같이 보여졌으며, 여러 개의 프로그램을 같이 실행되었을 때의 오류사항을 Java로 작성했던 것과 같이 동일하게 볼 수 있었습니다.

#!/bin/sh

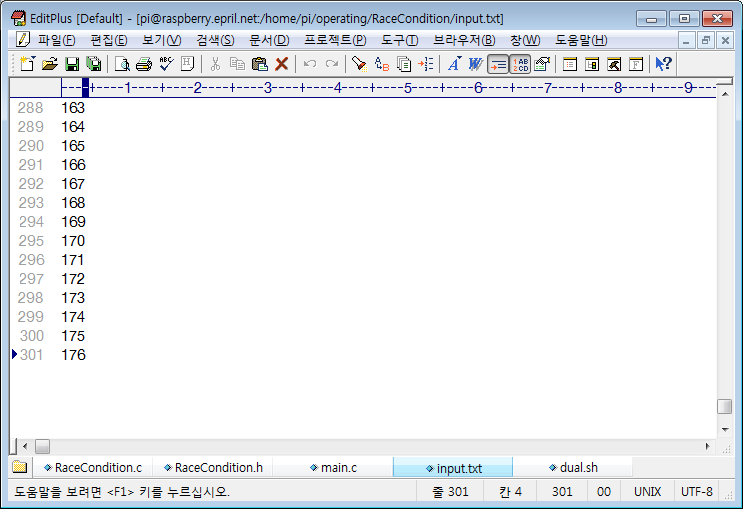
echo "Bash Script Start"

./main.exe &

./main.exe &

./main.exe

Code 6 dual.sh (Race Condition 실행)



Screen 2 결과 화면 (Race Condition)

제공해 주신 Advance IPC에서 발생되는 구문오류를 몇 가지 해결하고, C 코드로 구현한 RaceCondition을 include하고 Semaphore의 Lock으로서의 역할을 사용하여 자원 공유가 제대로 되는지 확인하였습니다.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <errno.h>

// Add library

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include "RaceCondition.h"

#define SEMPERM 0600

#define TRUE 1

#define FALSE 0

// Redefind Struct

typedef struct union\_semun {

int val;

struct semid\_ds \*buf;

ushort \*array;

} semun;

Code 7 improve.c (Use Semaphore)

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

int status = 0, semid;

if ( (semid = semget (semkey, 1, SEMPERM | IPC\_CREAT | IPC\_EXCL) ) == -1){

if ( (errno == EEXIST) )

semid = semget (semkey, 1, 0);

}else{

semun arg;

arg.val = n;

status = semctl(semid, 0, SETVAL, arg);

}

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 add1 (key\_t skey){

int semid;

pid\_t pid = getpid();

if ( (semid = initsem(skey,1) ) < 0)

exit(1);

printf("\nprocess %d before critical section\n", pid);

p(semid);

printf("process %d in critical section\n",pid);

addFile(); // 추가된 부분

printf("process %d leaving critical section\n", pid);

v(semid);

printf("process %d exiting\n",pid);

exit(0);

}

int main(){

key\_t semkey = 0x200;

int i;

for ( i=0; i<2; i++){

if (fork() == 0)

add1(semkey);

}

return 0;

}

Code 8 improve.c (Use Semaphore) cont.

역시 동일하게 **Code 9** Linux Shell을 이용하여 동시에 실행한 결과 **Screen 3** 와 같이 결과물을 얻을 수 있었습니다.

#!/bin/sh

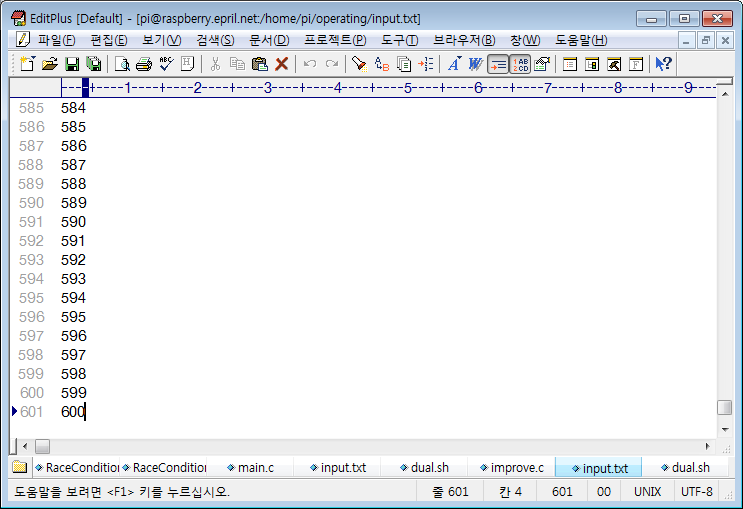
echo "Bash Script Start"

./improve.exe &

./improve.exe &

./improve.exe

Code 9 dual.sh (Improve Race Condition 실행)



Screen 3 결과 화면 (Improve Race Condition)

이것을 통하여 Semaphore가 Lock으로 작용하여 공유됨을 확인 할 수 있었습니다.

**3. 기타 : 개발환경, 실행화면**

개발환경은 RaspberryPi의 Raspbian Linux 운영체제를 이용하였습니다. Raspberry Pi 에 직접 모니터를 연결하여 작업한 것이 아닌 Window에서 Putty를 이용하여 접근하여 활용하였습니다. Putty 실행화면은 **Screen 4** 와 같으며 해당 화면에서 보이는 것과 같이 파일이 구성되어 있습니다.

컴파일의 경우 makefile을 만들어서 실행하였으며 내용은 **Code 10** 에 나와 있는 것과 같습니다.

improve.exe : improve.o RaceCondition.o

gcc -lm -o improve.exe improve.o RaceCondition.o

improve.o : improve.c

gcc -c improve.c

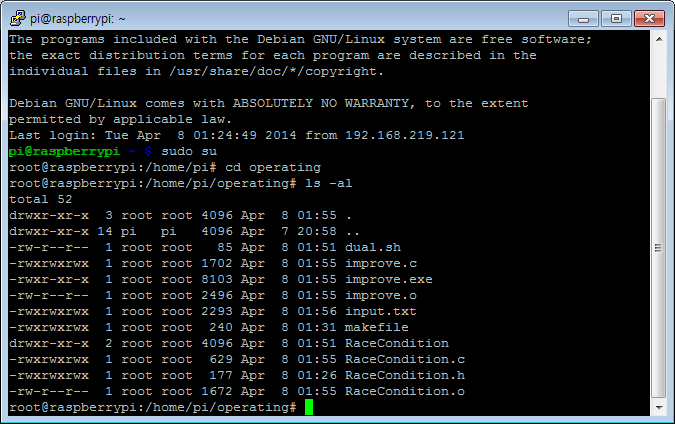
RaceCondition.o : RaceCondition.c RaceCondition.h

gcc -c RaceCondition.c

clean :

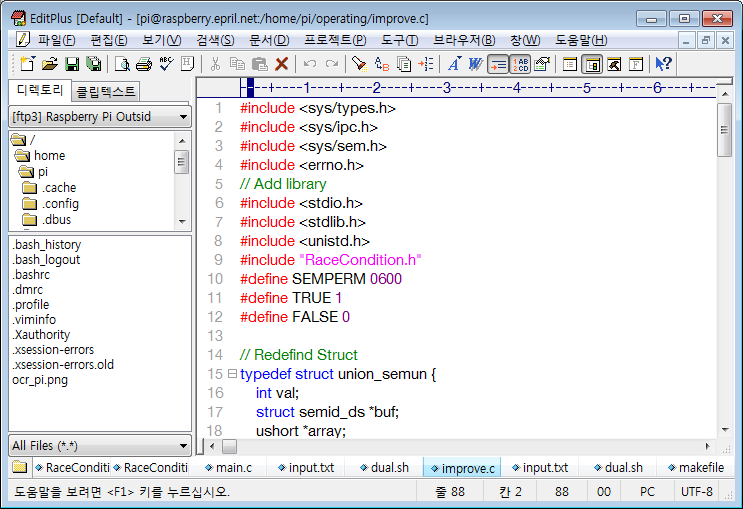
rm -f \*.exe

Code 10 makefile



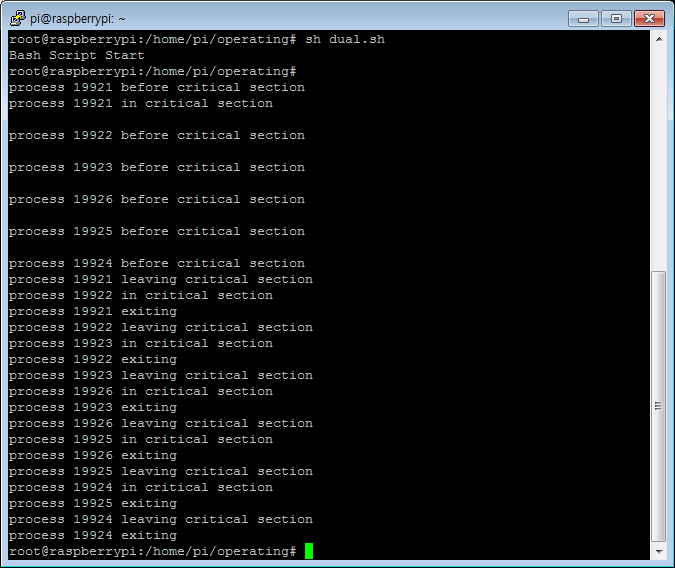
Screen 4 Putty operating folder list

파일 수정은 Editplus 프로그램을 FTP 접속을 이용하여 수행하였으며, 그 화면은 **Screen 5**와 같습니다.



Screen 5 Editplus

마지막으로 Linux Shell을 이용해서 실행된 화면은 **Screen 6**과 같습니다.



Screen 6 Linux Shell 실행화면