|  |
| --- |
| 멀티미디어 운영체제 |
| 과제2 – Lamport’s Bakery Algorithm 구현하기 |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| 과목 | 멀티미디어운영체제 |
| 담당교수 | 교수님 |
| 학과 | 멀티미디어공학과 |
| 학번 | 2015112614 |
| 이름 | 양동혁 |

이번 Lamport’s Bakery Algorithm 과제를 수행 하기 위해 C++ 언어를 기반으로 한 프로그래밍을 작성 하였고 코드 내 에서 프로그램이 어떻게 수행이 되고 있는지 얼만큼 진행이 되었는지 표시 하기 위해서 프로세스가 끝날 때 마다 몇 번째 프로세스가 몇 번째 들어 가게 되었는지 알 수 있도록 작성을 하게 되었습니다.

1. 소스 코드

#include <thread>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <Windows.h>

using namespace std;

#define PROCESSNUMBER 5

프로세스마다 기본적인 설정을 저장하는 구조체 입니다.

number 값은 프로세스의 고유 번호를 저장하고 time은 프로세스가 Critical Section에 얼마나 오랜 시간을 있을 것인지 랜덤으로 저장한 값, 마지막 stack 값은 프로세스가 몇 번째 프로세스에 들어 오게 되었는지 확인하는 값입니다.

typedef struct {

int number;

int time;

int stack;

}processStat;

bool choosing[PROCESSNUMBER] = { false };

int number[PROCESSNUMBER] = { 0 };

프로세스마다 기본적인 설정을 부여하는 값입니다.

프로세스마다 number 값을 지정해주고 프로세스마다 Critical Section 에 머무르는 시간을 입시간을 랜덤으로 0.1초 ~ 10초 사이로 지정해 주었습니다.

void MakeProcess(int \_i, processStat\*\* \_processInfo) {

\_processInfo[\_i] = (processStat\*)malloc(sizeof(processStat));

\_processInfo[\_i]->number = \_i;

\_processInfo[\_i]->time = (rand() % 100 + 10) \* 10;

\_processInfo[\_i]->stack = 0;

}

Lamport’s Barkey Algorithm 을 수행할 때 MaxNumber값을 반환해주는 함수입니다.

int MaxNumber() {

int maxNum = 0;

for (int i = 1; i < PROCESSNUMBER; i++)

if (number[maxNum] < number[i])

maxNum = i;

return number[maxNum];

}

Critical Section을 표현하기 위하여 임의의 함수를 만들었습니다.

Critical Section에 들어가게 된다면 이 함수에서 랜덤값으로 설정했던 time 만큼 잠깐 멈추고 함수를 탈출 하도록 실행하도록 했습니다.

void Critical\_Section(processStat\* processInfo) {

Sleep(processInfo->time);

processInfo->stack++;

}

Lamport’s Bakery Algorithm 중에서 (number[j],j) < (number[i],i)를 확인하는 함수입니다.

bool Check(int \_j, int \_i) {

if (number[\_j] < number[\_i])

return true;

else if (number[\_j] > number[\_i])

return false;

else {

if (\_j < \_i)

return true;

else

return false;

}

}

가장 중심이 되는 Lamport’s Bakery Algorithm입니다.

교과서 적인 기본 알고리즘을 토대로 하였으며 알고리즘이 모두 종료가 된다면 프로세스의 번호, 프로세스가 몇 번째 실행이 되었는지를 표시 하였고 또한 Critical Section 에서 걸린 시간과 Lamport’s Barkey 함수 자체 걸린 시간을 표시 할 수 있도록 표시 했습니다. 또한 프로세스 끝나고 다음 프로세스가 시작하기 전 까지 1 ~ 10초 사이로 계속 랜덤으로 지정 했습니다.

void LamportBakery(processStat \*processInfo) {

do {

clock\_t st = clock();

choosing[processInfo->number] = true;

number[processInfo->number] = MaxNumber() + 1;

choosing[processInfo->number] = false;

for (int j = 0; j < PROCESSNUMBER; j++) {

while (choosing[j]);

while ((number[j] != 0) && (Check(j, processInfo->number))); }

Critical\_Section(processInfo);

number[processInfo->number] = 0;

cout << processInfo->number << "번 프로세스 " << processInfo->stack << "번째 실행" << endl;

cout << "Critical Section " << float(processInfo->time) / 1000 << "초 걸림, 프로세스 대기시간 " << float(clock() - st) / 1000 << endl << endl;;

Sleep((rand() % 10 + 1) \* 1000);

} while (true);

}

void main() {

srand(time(NULL));

processStat\* processInfo[PROCESSNUMBER];

for (int i = 0; i < PROCESSNUMBER; i++)

MakeProcess(i, processInfo);

thread Process[10];

for (int i = 0; i < PROCESSNUMBER; i++) {

Process[i] = thread(LamportBakery, processInfo[i]);

Sleep(1000);

}

while (true);

}

Main 함수입니다. 가장 먼저 스레드의 상태를 프로세스의 수 만큼 만듭니다. 스레드 상태로는 스레드의 숫자, 시간, 횟수를 포함합니다. 그리고 스레드를 Lamport’s Barkey 에 1초를 주기로 실행을 시킵니다. Abort()를 자동으로 실행하지 않기 위하여 while(true)를 선언합니다.

void main() {

srand(time(NULL));

processStat\* processInfo[PROCESSNUMBER];

for (int i = 0; i < PROCESSNUMBER; i++)

MakeProcess(i, processInfo);

thread Process[10];

for (int i = 0; i < PROCESSNUMBER; i++) {

Process[i] = thread(LamportBakery, processInfo[i]);

Sleep(1000);

}

while (true);

}