Pthread를 이용하여 프로그램 구현하기

20172609

김시온

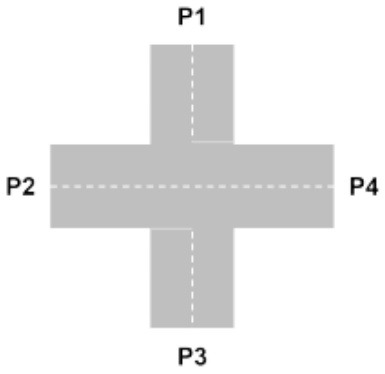
gcc 옵션 :

- gcc 20172609_OS6.c -pthread

Makefile 없음.

- 서론

Pthread를 이용하여 교차로에서의 차량 진입 문제 해결 프로그램 구현



명세서에 주어진 조건대로 주어진 프로그램을 완성한다.

- 본론

결과

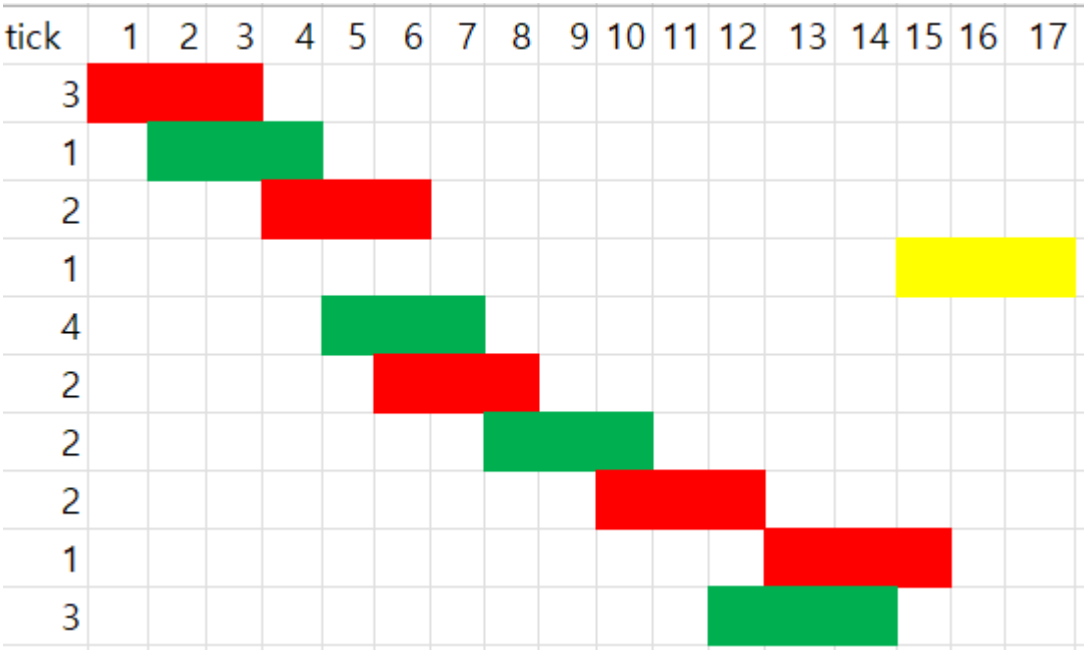
```
Total number of vehicles : 10
Start point : 3 1 2 1 4 2 2 2 1 3
tick: 1
==============================
Passed Vehicle
Car
Waiting Vehicle
Car
==============================
tick: 2
==============================
Passed Vehicle
Car 3
Waiting Vehicle
Car
==============================
tick: 3
==============================
Passed Vehicle
Car 1
Waiting Vehicle
Car 2
==============================
tick: 4
==============================
Passed Vehicle
Car
Waiting Vehicle
Car 1
==============================
tick: 5
==============================
Passed Vehicle
Car 2
Waiting Vehicle
Car 1
==============================
tick: 6
==============================
Passed Vehicle
Car 4
Waiting Vehicle
Car 1
```

```
==============================
tick: 7
==============================
Passed Vehicle
Car 2
Waiting Vehicle
Car 1 2
==============================
tick: 8
==============================
Passed Vehicle
Car
Waiting Vehicle
Car 1 2
==============================
tick: 9
==============================
Passed Vehicle
Car 2
Waiting Vehicle
Car 1 1 2
==============================
tick: 10
==============================
Passed Vehicle
Car
Waiting Vehicle
Car 1 1 3
==============================
tick: 11
==============================
Passed Vehicle
Car 2
Waiting Vehicle
Car 1 1 3
==============================
tick: 12
==============================
Passed Vehicle
Car
Waiting Vehicle
Car 1 1
```

```
================================
tick: 13
================================
Passed Vehicle
Car 3
Waiting Vehicle
Car 1
================================
tick: 14
================================
Passed Vehicle
Car 1
Waiting Vehicle
Car 1
================================
tick: 15
================================
Passed Vehicle
Car
Waiting Vehicle
Car
================================
tick: 16
================================
Passed Vehicle
Car 1
Waiting Vehicle
Car
================================
tick: 17
================================
Passed Vehicle
Car
Waiting Vehicle
Car
================================
Number of vehicles passed from each start point
P1 : 3 times
P2 : 4 times
P3 : 2 times
P4 : 1 times
Total time : 17 ticks
```

위에 상황은 3 1 2 1 4 2 2 2 1 3 차로로 차가 들어올 때이고,

이를 표로 만들면,



주어진 조건을 만족하며 차가 운행됨을 알 수 있습니다.

(가로로 시간순, 세로는 들어온 차의 표입니다.)

예를 들어 1~2.999초동안 운행된 차는 처음에 3번째 도로로 들어온 차입니다.

- 전체 소스코드

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <stdbool.h>

void *ssu_thread1(void *arg);
void *ssu_thread2(void *arg);
void *ssu_thread3(void *arg);
void *ssu_thread4(void *arg);

pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex2 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex3 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex4 = PTHREAD_MUTEX_INITIALIZER;

pthread_cond_t cond1 = PTHREAD_COND_INITIALIZER;
pthread_cond_t cond2 = PTHREAD_COND_INITIALIZER;
pthread_cond_t cond3 = PTHREAD_COND_INITIALIZER;
pthread_cond_t cond4 = PTHREAD_COND_INITIALIZER;
pthread_cond_t cond5 = PTHREAD_COND_INITIALIZER; //끝나면 보내는

int arr[16];
int N;
int tick = 1;

int f1, f2, f3, f4;
int wait[5];
int p[5];

void input()
{
    printf("Total number of vehicles : ");
    scanf("%d", &N);
    if (N < 10 || N > 15)
    {
        printf("10 ~ 15 사이의 숫자를 입력해주세요.\n");
        exit(1);
    }
    srand(time(NULL));
    for (int i = 0; i < N; i++)
        arr[i] = rand() % 4 + 1;
    printf("Start point : ");
    for (int i = 0; i < N; i++)
```

```
        printf("%d ", arr[i]);
    printf("\n");
}

pthread_cond_t *func(int n)
{
    switch (n)
    {
    case 1:
        return &cond1;
    case 2:
        return &cond2;
    case 3:
        return &cond3;
    case 4:
        return &cond4;
    default:
        printf("error\n");
        exit(-1);
        return NULL;
    }
}
int Passed;
void printTick()
{
    printf("tick: %d\n", tick);
    printf("================================\n");
}
void print_Waiting()
{
    int tmp[5] = {0, 0, 0, 0, 0};
    for (int i = 1; i < 5; i++)
    {
        if (wait[i] == 0)
            continue;
        else
        {
            tmp[i] = wait[i];
        }
    }

    for (int i = 1; i < 5; i++)
    {
        while (tmp[i] > 0)
        {
            printf("%d ", i);
            tmp[i]--;
        }
```

```
    }
}
void print_Vehicle()
{
    printf("Passed Vehicle\nCar ");
    if (1 <= Passed && Passed <= 4)
    {
        printf("%d", Passed);
        p[Passed]++;
    }
    printf("\n");
    printf("Waiting Vehicle\nCar ");
    print_Waiting();
    printf("\n");
    printf("================================\n");
    Passed = 0;
}

void Print()
{
    printTick();
    print_Vehicle();
    if (tick < N) // tick이 N보다 작으면{
    {
        wait[arr[tick]]++;
    }
    tick++;
}

bool Is_Waiting(int n)
{
    if (wait[n] > 0)
        return true;
    return false;
}

int Pick_In_Waiting()
{
    int cnt = 0;
    for (int i = 1; i <= 4; i++)
    {
        if (Is_Waiting(i) == 0)
            cnt++;
    }
    if (cnt == 4) //다 없을 때
    {
        if (Passed == 0)
        {
```

```
            Print();
            tick--;
            printf("Number of vehicles passed from each start point\n");
            for (int i = 1; i <= 4; i++)
                printf("P%d : %d times\n", i, p[i]);
            printf("Total time : %d ticks\n", tick);
            exit(0);
        }
        return 0;
    }

    srand(time(NULL));
    int n = rand() % 4 + 1;
    while (1)
    {
        if (Is_Waiting(n)) //있으면 뽑음
            return n;
        srand(time(NULL));
        n = rand() % 4 + 1;
    }
}
int main()
{
    pthread_t tid1, tid2, tid3, tid4;
    int status;

    if (pthread_create(&tid1, NULL, ssu_thread1, NULL) != 0)
    {
        fprintf(stderr, "pthread_create error\n");
        exit(1);
    }
    if (pthread_create(&tid2, NULL, ssu_thread2, NULL) != 0)
    {
        fprintf(stderr, "pthread_create error\n");
        exit(1);
    }
    if (pthread_create(&tid3, NULL, ssu_thread3, NULL) != 0)
    {
        fprintf(stderr, "pthread_create error\n");
        exit(1);
    }
    if (pthread_create(&tid4, NULL, ssu_thread4, NULL) != 0)
    {
        fprintf(stderr, "pthread_create error\n");
        exit(1);
    }
    sleep(1);
    input();
```

```
    wait[arr[0]]++;
    pthread_cond_signal(func(arr[0])); //처음 신호 보내기

    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    pthread_join(tid3, NULL);
    pthread_join(tid4, NULL);

    printf("Number of vehicles passed from each start point\n");
    for (int i = 1; i <= 4; i++)
        printf("P%d : %d times\n", i, p[i]);
    printf("Total time : %d ticks\n", tick);
    return 0;
}

void *ssu_thread1(void *arg)
{
    while (1)
    {
    ONE:
        pthread_mutex_lock(&mutex1);
        pthread_cond_wait(&cond1, &mutex1);
    ONE2:
        if (wait[1] > 0) //차 있을 경우
        {
            //    pthread_cond_wait(&cond1, &mutex1);
            wait[1]--;
            Print();
            usleep(1000000); // 1 이 일을 하고
            Passed = 1;

            if (Is_Waiting(3))
            { //반대편이 있으면 그 애 출발시키고
                pthread_cond_signal(&cond3);
                // 0.99 초후 자신이 도착인거 보냄.
                pthread_mutex_unlock(&mutex1);
                goto ONE; //처음으로 돌아감 ? or break;
            }
            else
            { // 1 초 지나고 출발할 친구가 없음 ( 맞은편이 없기때문에 )
                Print();
            }
            usleep(1000000); // 0.99 초 지나고

            int next = Pick_In_Waiting(); //다음꺼 랜덤뽑기

            if (next != 0) //기다리는게 있다면
            {               //자기자신이면
```

```
                if (next == 1)
                {
                    goto ONE2;
                }
                pthread_cond_signal(func(next)); //다음 쓰레드에게 signal 줌
            }
        }
        // wait[1]이 0 일때  (안기다리는데 호출을 줄 때 :) else
        pthread_mutex_unlock(&mutex1);
    }
    return NULL;
}
void *ssu_thread2(void *arg)
{
    while (1)
    {
    TWO:
        pthread_mutex_lock(&mutex2);
        pthread_cond_wait(&cond2, &mutex2);
    TWO2:
        if (wait[2] > 0) //차 있을 경우
        {
            //    pthread_cond_wait(&cond1, &mutex1);
            wait[2]--;
            Print();
            usleep(1000000); // 1 이 일을 하고
            Passed = 2;
            if (Is_Waiting(4))
            { //반대편이 있으면 그 애 출발시키고
                pthread_cond_signal(&cond4);
                pthread_mutex_unlock(&mutex2);
                goto TWO; //처음으로 돌아감 ? or break;
            }
            else
            { // 1 초 지나고 출발할 친구가 없음 ( 맞은편이 없기때문에 )
                Print();
            }
            usleep(1000000); // 0.99 초 지나고

            int next = Pick_In_Waiting(); //다음꺼 랜덤뽑기
            if (next != 0)                //기다리는게 있다면
            {                             //자기자신이면
                if (next == 2)
                {
                    goto TWO2;
                }
                pthread_cond_signal(func(next)); //다음 쓰레드에게 signal 줌
            }
```

```
        }
        pthread_mutex_unlock(&mutex2);
    }
    return NULL;
}
void *ssu_thread3(void *arg)
{
    while (1)
    {
    THREE:
        pthread_mutex_lock(&mutex3);
        pthread_cond_wait(&cond3, &mutex3);
    THREE2:
        if (wait[3] > 0) //차 있을 경우
        {
            //    pthread_cond_wait(&cond1, &mutex1);
            wait[3]--;
            Print();
            usleep(1000000); // 1 이 일을 하고
            Passed = 3;

            if (Is_Waiting(1))
            { //반대편이 있으면 그 애 출발시키고
                pthread_cond_signal(&cond1);
                // 0.99 초후 자신이 도착인거 보냄.
                pthread_mutex_unlock(&mutex3);
                goto THREE; //처음으로 돌아감 ? or break;
            }
            else
            { // 1 초 지나고 출발할 친구가 없음 ( 맞은편이 없기때문에 )
                Print();
            }
            usleep(1000000); // 0.99 초 지나고

            int next = Pick_In_Waiting(); //다음꺼 랜덤뽑기
            if (next != 0)                //기다리는게 있다면
            {                             //자기자신이면
                if (next == 3)
                {
                    goto THREE2;
                }
                pthread_cond_signal(func(next)); //다음 쓰레드에게 signal 줌
            }
        }
        // wait[1]이 0 일때  (안 r 다리는데 호출을 줄 때 :) else
        pthread_mutex_unlock(&mutex3);
    }
    return NULL;
```

```
}
void *ssu_thread4(void *arg)
{
    while (1)
    {
    FOUR:
        pthread_mutex_lock(&mutex4);
        pthread_cond_wait(&cond4, &mutex4);
    FOUR2:
        if (wait[4] > 0) //차 있을 경우
        {
            //    pthread_cond_wait(&cond1, &mutex1);
            wait[4]--;
            Print();
            usleep(1000000); // 1 이 일을 하고
            Passed = 4;

            if (Is_Waiting(2))
            { //반대편이 있으면 그 애 출발시키고
                pthread_cond_signal(&cond2);
                // 0.99 초후 자신이 도착인거 보냄.
                pthread_mutex_unlock(&mutex4);
                goto FOUR; //처음으로 돌아감 ? or break;
            }
            else
            { // 1 초 지나고 출발할 친구가 없음 ( 맞은편이 없기때문에 )
                Print();
            }
            usleep(1000000); // 0.99 초 지나고

            int next = Pick_In_Waiting(); //다음꺼 랜덤뽑기
            if (next != 0)                //기다리는게 있다면
            {                             //자기자신이면
                if (next == 4)
                {
                    goto FOUR2;
                }
                pthread_cond_signal(func(next)); //다음 쓰레드에게 signal 줌
            }
        }
        // wait[1]이 0 일때  (안기다리는데 호출을 줄 때 :) else
        pthread_mutex_unlock(&mutex4);
    }
    return NULL;
}
```

- 결론 및 느낀점

c언어에서 pthread 함수를 통해 멀티쓰레드로 코드를 구현하였다.

리눅스 시스템 프로그래밍을 수강하지 않았던 나로서 굉장히 까다롭고 어려웠던 과제였다.

왜 동기화가 필요한지를 절실히 깨달았고,

같은 코드여도 다른결과값이 나올 수가 있다를 몸소 체험한 과제였다.

앞으로 멀티쓰레드, 멀티 프로세싱 작업을 할 경우, 여러가지를 생각하고 차분히 코드를 작성하는 버릇을 가져야겠다고 생각했다.