**REPORT**

**| Operating Systems**

**| Multi-threaded Programming**

**| 컴퓨터정보공학과**

**| 12141540 박영창**

# Objective

Operating system 수업 시간에 배웠던 Multi-thread를 활용해 multi-threaded program을 구현해 보는 것이 목적이었습니다.

Linux 운영체제를 이용해 program을 실행하는 것만 해도 어려웠는데, virtual box를 통해 Linux를 사용하니 속도가 너무 느려서 구름IDE라는 사이트를 사용해 Linux 환경에서 컴파일 및 program 실행을 했습니다.

지금까지 만들었던 program에서는 single-thread 방식만 사용했지만, 이번 과제를 통해 multi-thread 방식을 사용하면서 thread 생성, thread에 작업을 부여하는 것에 대해 알 수 있었습니다.

# First Program

**개요**

첫 번째 program은 txt파일을 통해 81개의 수를 9x9 크기의 table에 입력 받고, 이를 3x3 크기의 subgrid들로 나눠 그 안에서의 최대값을 구하는 9개의 selection thread를 생성, 실행하고 9개의 thread의 결과값들 중에서 최대값을 구하는 parent thread를 생성, 실행해 최대값을 구하는 program이었습니다.

**변수**

* Int arr[9][9] | 파일로부터 입력 받은 값이 모두 저장되는 배열
* Int res[9] | selection thread의 결과값이 저장되는 배열
* Int resIndex | selection thread가 res 배열에 접근하기 위한 인덱스
* Int Result | 최종 결과값
* Struct index | subgrid의 시작점의 x, y좌표가 들어있는 구조체
* Index param[9] | 9개의 selection thread에 넘겨줄 index 구조체 배열
* Pthread\_t tid[10] | 생성될 10개의 thread의 thread id

**함수**

* Void setIndex(index \*p, int \_x, int \_y)

인자로 넘겨받은 index 구조체 p의 x, y좌표를 인자로 받은 \_x, \_y로 설정

* Void \*highestSubgrids(void \*dt) | selection thread

인자로는 index구조체를 받으며 입력 받은 구조체의 x, y좌표를 기준으로 3x3 크기의 subgrid를 확인해 가장 높은 값을 res배열에 저장

* Void \*highestResult() | parent thread

res배열 내의 가장 큰 값을 Result에 저장

**실행**

1. input.txt라는 파일을 불러와 안에 있는 값들을 arr 배열에 저장합니다.
2. 구조체 배열 param[9]에 각각 subgrid들의 시작점의 좌표를 저장합니다.
3. Pthread\_create를 통해 selection thread를 2.에서 저장한 param[9] 구조체를 인자로써 실행시킵니다.
4. Pthread\_join을 통해 selection thread가 모두 끝나기를 기다립니다.
5. Pthread\_create를 통해 parent thread를 실행시키고, pthread\_join을 통해 끝나기를 기다립니다.
6. 결과값인 Result를 출력하고, 파일 입출력을 종료합니다.

**결과**

함께 첨부한 highest.jpg로 보았을 때, testcase 1은 1부터 81까지 순서대로 저장되어 있는 txt파일을 사용했고, testcase 2는 testcase 1의 파일에서 중간의 아무 값이나 99로 변경한 파일을 사용했습니다. 출력 결과로는 txt 파일 내의 최대값인 81과 99가 정상적으로 출력되었습니다.

# Second Program

**개요**

두 번째 program은 txt파일을 통해 수들을 입력 받고, 이를 3개의 sorting thread가 구간을 나눠서 정렬한 뒤에 그 구간의 최소값들을 merging thread가 확인해 전체의 최소값을 찾는 program이었습니다. 문제의 조건은 Merging thread에서 spin lock을 통해 sorting thread의 종료까지 merging thread는 작업을 수행하지 않도록 만드는 것이었습니다.

**변수**

* Int arr[100] | 파일로부터 입력 받은 값이 모두 저장되는 배열
* Int result[3] | sorting thread의 결과값이 저장되는 배열
* Int lowestValue | 최종 결과값
* Int resultIndex | selection thread에서 result 배열에 접근할 인덱스
* Struct index | 나눠진 구간의 시작과 끝, spin lock을 위한 check

변수가 있는 구조체

* Index param[3] | 3개의 sorting thread에 넘겨줄 index 구조체 배열
* Pthread\_t tid[4] | 생성될 4개의 thread의 thread id

**함수**

* Void \*sorting(void \*\_p) | sorting thread

인자로는 index구조체를 받으며 입력 받은 구조체의 start, end를 기준으로 구간을 나눠 해당 구간을 정렬하고, 가장 작은 값을 result 배열에 저장. 함수가 완료될 때 입력 받은 구조체의 check 변수를 1로 변경

* Void \*merge() | merging thread

3 개의 sorting thread가 종료되었는지 spin-lock을 통해 확인 ( 구조체 배열 모든 인덱스의 check 변수가 1로 바뀌었는지 확인 ), 모두 종료되었다면 result배열에서 최소값을 찾아 lowestValue에 저장

**실행**

1. input.txt라는 파일을 불러와 안에 있는 값들을 arr 배열에 저장합니다.
2. 구조체 배열 param[3]에 각 구간의 시작점과 끝점을 저장합니다.
3. Pthread\_create를 통해 sorting thread를 2.에서 저장한 param[3] 구조체를 인자로써 실행시킵니다. 동시에 merging thread도 실행시킵니다.
4. Sorting thread와 merging thread가 동시에 수행하지만, sorting thread가 모두 끝나기 전까지는 merging thread는 spin-lock 때문에 멈춰 있는 상태입니다. Sorting thread가 모두 끝나면, merging thread를 작동시킵니다.
5. Pthread\_join을 통해 main에서 thread가 모두 끝나기를 기다립니다.
6. 결과값인 lowestValue를 출력하고, 파일 입출력을 종료합니다.

**결과**

함께 첨부한 lowest.jpg로 보았을 때, testcase 1은 과제 pdf파일로 주어진 수들과 같은 수로 입력되어 있는 input.txt 파일을 사용했고, testcase 2는 중간에 testcase 1의 최소값인 2보다 작은 1을 저장한 파일을 사용해 보았습니다. 출력 결과로는 각각 testcase의 최소값인 2, 1이 맞게 출력되었습니다.