오퍼레이팅시스템 – 1st Homework

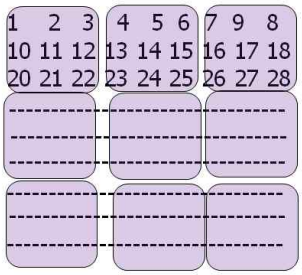
12131819 육동현

1 번 문항 – 최댓값 구하기

A_문제설명

9 개의 스레드는 3X3 Subgrid 를 전달받아 각각이 구한 최댓값을 10 번째 스레드에게 전달함.
10 번째 스레드(selection)는 9 개의 값 중 최댓값을 구함.
(단, input.txt 파일에는 중복되는 숫자가 없다고 가정함)

input.txt



B_코드설명

```
/* 파일 읽어들이는 처리 */
FILE *fp = NULL;
fp = fopen("input.txt", "r");
int** input = (int **)malloc(sizeof(int *) * TNUM);
for (i = 0; i < TNUM; i++)
    input[i] = (int *)malloc(sizeof(int) * TNUM);
share = (int **)malloc(sizeof(int *) * TNUM);
for (i = 0; i < TNUM; i++)
    share[i] = (int *)malloc(sizeof(int) * TNUM);
for (i = 0; i < TNUM; i++)
{
    for (j = 0; j < TNUM; j++)
    {
        fscanf(fp, "%d ", &input[i][j]);
        share[i][j] = 0;
    }
}
buf* bf = (buf*)malloc(sizeof(buf));
bf->in = input;
```

우선 input.txt 파일로부터 자료를 9X9 형태로 읽어 들임.

```
/* 읽어온 자료를 각 스레드에게 Grid로 할당하기 위해 slicing하는 작업 */
for (i = 0; i < GSIZE; i++)
{
    for (j = 0; j < GSIZE; j++)
    {
        a0[i][j] = input[i][j];
        a1[i][j] = input[i][j + 3];
        a2[i][j] = input[i][j + 6];
        a3[i][j] = input[i + 3][j];
        a4[i][j] = input[i + 3][j + 3];
        a5[i][j] = input[i + 3][j + 6];
        a6[i][j] = input[i + 6][j];
        a7[i][j] = input[i + 6][j + 3];
        a8[i][j] = input[i + 6][j + 6];
    }
}
```

앞에서 읽어 들인 데이터를 각각의 Subgrid 에게 3X3 형식으로 할당하기 위해서 데이터를 가공,
즉 slicing 하는 작업을 수행함.

```
116     /* 자식 스레드의 생성 */
117     pthread_create(&id_t[0], NULL, maximum, (void *)s0);
118     pthread_create(&id_t[1], NULL, maximum, (void *)s1);
119     pthread_create(&id_t[2], NULL, maximum, (void *)s2);
120     pthread_create(&id_t[3], NULL, maximum, (void *)s3);
121     pthread_create(&id_t[4], NULL, maximum, (void *)s4);
122     pthread_create(&id_t[5], NULL, maximum, (void *)s5);
123     pthread_create(&id_t[6], NULL, maximum, (void *)s6);
124     pthread_create(&id_t[7], NULL, maximum, (void *)s7);
125     pthread_create(&id_t[8], NULL, maximum, (void *)s8);
126
```

pthread_create 함수를 통해서 자식스레드를 생성함. 첫 번째 인자는 스레드의 ID 를 의미하고, 두 번째 인자는 스레드의 특성을 의미하나 우리는 여기서 NULL 을 집어 넣음, 세 번째 인자는 실행할 함수를 의미하고, 마지막 인자는 앞의 함수에 전달할 인자를 의미함.

```
127     /* 자식스레드를 reap하여 최종 selection을 하는 처리 */
128     min* mini = (min*)malloc(sizeof(min));
129     mini->m = share;
130     for (i = 0; i < 10; i++)
131         pthread_join(id_t[i], &t_return[i]); // 자식 스레드로부터 reaping
132     fin_result = (int)t_return[0];
133     for (i = 0; i < TNUM; i++)
134     {
135         if (fin_result < t_return[i])
136             fin_result = t_return[i];
137     } printf("\nSelection thread # %lx's maximum : %d\n", pthread_self(), fin_result);
```

이 부분에서는 자식스레드를 pthread_join 함수를 통해 reap 하면서 반환 받은 결과를 이용하여 9 개의 스레드가 반환한 값 중 가장 큰 값을 구함. 즉, 여기에서 main 스레드가 selection thread 의 역할을 수행함.

```
142 void* maximum(void *arg)
143 {
144     int result; // 스레드별 결과를 저장하는 변수
145     int i, j; // 제어변수
146     int tn = 0; // 스레드의 번호
147     slice sl = *(slice*)arg;
148     for (i = 0; i < GSIZE; i++)
149         for (j = 0; j < GSIZE; j++)
150             if (result < sl.element[i][j])
151                 result = sl.element[i][j];
152     share[0][tn++] = result;
153     printf("Child thread # %lx's maximum : %d\n", pthread_self(), result);
154     pthread_exit((void*)result);
155 }
```

자식스레드는 처리한 결과를 pthread_exit 함수를 통해서 부모스레드에게 넘겨줌.
여기서 pthread_self 함수는 실행하고 있는 스레드의 ID 를 나타내는 함수이다.

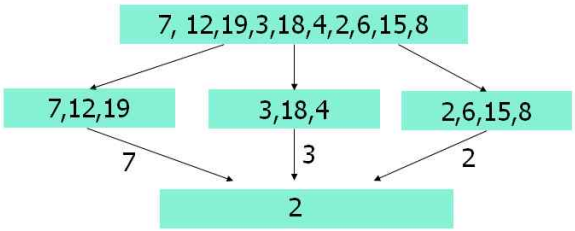
C_실행결과

```
Aomori Kaitou@DESKTOP-KAITOU ~/os1
$ ./hw1.exe
These are the data from input.txt :

  1   2   3   4   5   6   7   9   8
 10  11  12  13  14  15  16  17  18
 20  21  22  23  24  25  26  27  28
 30  31  32  33  34  35  36  37  38
 40  41  42  43  44  45  46  47  48
 50  51  52  53  54  55  56  57  58
 60  61  62  63  64  65  66  67  68
 70  71  72  73  74  75  76  77  78
 80  81  82  83  84  85  86  87  88

Child thread # 6000591b0's maximum : 22
Child thread # 6000592b0's maximum : 25
Child thread # 6000593b0's maximum : 28
Child thread # 6000594b0's maximum : 52
Child thread # 6000595b0's maximum : 55
Child thread # 6000596b0's maximum : 58
Child thread # 6000597b0's maximum : 82
Child thread # 6000598b0's maximum : 85
Child thread # 6000599b0's maximum : 88

Selection thread # 600000010's maximum : 88
```

2 번 문항 – 최솟값 구하기

A_문제설명

전역배열은 각 스레드에 의해 공유됨. 각각의 스레드에게 시작과 끝 인덱스 번호를 전달해야 함.
최종적으로 결과를 받는 스레드는 나머지 3 개의 스레드가 결과를 줄 때까지 스핀락을 수행함.



B_코드설명

```
8
9   /* 전역변수 */
10  int arr[10]; // 전역배열
11  int part1[2] = { 0, 2 }; // 첫번째 조각
12  int part2[2] = { 3, 5 }; // 두번째 조각
13  int part3[2] = { 6, 9 }; // 세번째 조각
14
```

"Input.txt" 파일에서 받아온 데이터를 전역 배열 arr 에 넣어 모든 스레드가 동시에 접근할 수 있도록 함. 즉, 전역배열은 모든 스레드가 접근할 수 있는 공유영역이 됨. 이 사례에서는 전역배열을 단지 읽어 들이기만 하기 때문에 임계영역으로 설정하지 않아도 큰 문제가 없었음. 또한 11~13 번째 라인은 각 데이터를 3 조각으로 자르는 데 필요한 시작 인덱스번호와 끝 인덱스 번호를 나타냄.

```
/* 자식스레드 생성 */
pthread_create(&id_t[0], NULL, minimum, (void*)part1);
pthread_create(&id_t[1], NULL, minimum, (void*)part2);
pthread_create(&id_t[2], NULL, minimum, (void*)part3);
```

pthread_create 함수를 통해서 자식스레드를 생성함.
첫번째 인자는 스레드의 ID 를 의미하고, 두번째 인자는 attr 를 의미하며, 세번째 인자는 실행할 함수를 의미하고, 마지막 인자는 앞의 함수에 전달할 인자를 의미함.

```
/* 자식스레드 거두기 */
pthread_join(id_t[0], &t_return[0]);
pthread_join(id_t[1], &t_return[1]);
pthread_join(id_t[2], &t_return[2]);
```

pthread_join 함수를 호출하게 되면 부모스레드는 자식스레드가 처리를 마칠 때까지 block 상태로 대기한다.

```
5   #define TRUE 1 // 실행된 상태
6   #define FALSE 0 // 실행안된 상태
7   int turn = 0; // 호출된 스레드 개수 카운트
8   int status[TNUM] = { FALSE }; // 초기에는 모든 스레드가 실행 안 된 상태
```

스핀락을 구현하기 위해 실행된 상태와 실행 안 된 상태를 판단하는 배열 status 를 생성
status 는 각 스레드의 완료여부를 체크하는 역할을 수행함.
여기서 TRUE 는 실행된 상태를, FALSE 는 실행되지 않은 상태를 나타냄

```
42
43      do // 스핀락 메커니즘의 구현
44      {
45          if (status[0] && status[1] && status[2]) // 세 자식 스레드의 수행이 모두 완료되어야만 진입가능
46          {
47              /* the process to get minimum value from threads */
48              fin_result = (int)t_return[0]; // 자식스레드에서 받아온 임의의 값으로 초기화
49              for (i = 0; i < 3; i++)
50                  if (fin_result >(int)t_return[i]) // 만약 초기화된 값이 자식스레드의 리턴값보다 작으면,
51                      fin_result = (int)t_return[i]; // fin_result의 값이 해당 값으로 갱신됨
52              printf("\nChild thread # %lx's minimum : %d\n", id_t[0], (int)t_return[0]);
53              printf("Child thread # %lx's minimum : %d\n", id_t[1], (int)t_return[1]);
54              printf("Child thread # %lx's minimum : %d\n", id_t[2], (int)t_return[2]);
55              printf("\nParent thread # %lx's minimum : %d\n", pthread_self(), fin_result);
56              break;
57          } wait_count++;
58          printf("waiting count : %d\n", wait_count);
59      } while (TRUE);
60
61      return EXIT_SUCCESS;
62  }
```

45 번째 라인은 논리곱연산(AND)를 이용하여 모든 스레드가 다 실행되어야 TRUE 가 되어 if 블록 내부로 진입하게 하고 그렇지 않은 경우 do while 루프를 돌면서 대기하게 됨.

```
64  void* minimum(void* arg)
65  {
66      int result; // 각 자식스레드의 결과값을 저장하는 변수
67      int start = ((int*)arg)[0]; // 시작 인덱스 번호
68      int end = ((int*)arg)[1]; // 종료 인덱스 번호
69
70      // 전역배열로부터 최솟값을 구하는 처리
71      result = arr[start]; // 전역배열의 임의의 값으로 초기화
72      int i; // 제어변수
73      for (i = start; i <= end; i++)
74      {
75          if (result > arr[i])
76              result = arr[i];
77          printf("tid # %lx : the minimum number is %d\n", pthread_self(), result);
78      }
79      status[turn] = TRUE; // 자식스레드의 처리가 끝났음을 알림
80      turn++; // 스레드 번호를 갱신
81
82      pthread_exit((void*)result);
83  }
```

start 와 end 는 pthread_create 의 네 번째 인자를 통해 값을 전달 받음. 이를 통해 조각의 시작과 끝 인덱스 번호를 설정하고 이에 따라 해당 조각의 최솟값을 구하는 처리를 수행함. 또한 pthread_exit 을 통해 pthread_join 함수에 값을 전달함.

C_실행결과

```
Aomori Kaitou@DESKTOP-KAITOU ~/os2
$ ./hw2.exe
tid # 600048790 : the minimum number is 7
tid # 600048890 : the minimum number is 3
tid # 600048790 : the minimum number is 7
tid # 600080b00 : the minimum number is 2
tid # 600048890 : the minimum number is 3
tid # 600048790 : the minimum number is 7
tid # 600080b00 : the minimum number is 2
tid # 600048890 : the minimum number is 3
tid # 600080b00 : the minimum number is 2
tid # 600080b00 : the minimum number is 2

Child thread # 600048790's minimum : 7
Child thread # 600048890's minimum : 3
Child thread # 600080b00's minimum : 2

Parent thread # 600000010's minimum : 2
```