

Project#2 Sudoku

2020062324 이은비

1. 함수 설명

void *check_rows(void *arg)

```
int i, j;

for (i = 0; i < 9; i++) {
    int check[10] = {}; /* 해당 행에 1~9까지의 숫자가 있는지 확인하는 배열.
                        숫자 n이 있다면 n번 인덱스의 값은 1, 아니면 0이 된다. (0번 인덱스는 사용하지 않음) */
    for (j = 0; j < 9; j++) {
        check[sudoku[i][j]] = 1; /* sudoku[i][j]의 값(숫자)을 check 배열의 인덱스로 하여 해당 값을 1로 바꾼다. */
    }

    for (j = 1; j < 10 && check[j]; j++); /* check의 1~9번 인덱스의 값이 0일때까지 계속 반복.
                                           만약 모두 1이라면 j는 10이 되어 종료됨. */
    j == 10 ? (valid[0][i] = 1) : (valid[0][i] = 0); /* 만약 j가 10이라면(check가 모두 1이라면)
                                                       valid[0][i]의 값을 1로, 아니면 0으로 한다. */
}
pthread_exit(NULL);
```

		j								
i		0	1	2	3	4	5	6	7	8
0		6	3	9	8	4	1	2	7	5
1		7	2	4	9	5	3	1	6	8
2		1	8	5	7	2	6	3	9	4
3		2	5	6	1	3	7	4	8	9
4		4	9	1	5	8	2	6	3	7
5		8	7	3	2	6	9	5	2	1
6		5	4	4	3	9	8	7	1	6
7		3	1	8	6	7	5	9	4	2
8		9	6	7	2	1	4	8	5	3

		i = 4								
		4	9	1	5	8	2	6	3	7
check[]	index	1	2	3	4	5	6	7	8	9
		1	1	1	1	1	1	1	1	1
		i = 5								
		8	7	3	2	6	9	5	2	1
check[]	index	1	2	3	4	5	6	7	8	9
		1	1	1	0	1	1	1	1	1
										⇒ valid[0][4] = 1
										⇒ valid[0][5] = 0

변수 i는 sudoku의 행 번호, j는 sudoku의 열을 나타낸다. check[n]은 숫자 n이 배열 안에 있는지 여부를 저장하는 배열이다. sudoku[i][j]의 값(숫자)을 인덱스로 하여 check[sudoku[i][j]]를 1로 할당한다. 만약 check[1]부터 check[9]까지 값이 모두 1 이라면 숫자 1~9가 해당 행에 모두 있다는 뜻이다. 만약 check[1]~check[9] 중 하나라도 1이 아닌 값이 있다면 유효하지 않다.

void *check_columns(void *arg)

```

int i, j;

for (i = 0; i < 9; i++) {
    int check[10] = {}; /* 해당 열에 1~9까지의 숫자가 있는지 확인하는 배열.
                        숫자 n이 있다면 n번 인덱스의 값은 1, 아니라면 0이 된다. (0번 인덱스는 사용하지 않음) */

    for (j = 0; j < 9; j++) {
        check[sudoku[j][i]] = 1; /* sudoku[j][i]의 값(숫자)을 check 배열의 인덱스로 하여 해당 값을 1로 바꾼다. */
    }

    for (j = 1; j < 10 && check[j]; j++); /* check의 1~9번 인덱스의 값이 0일때까지 계속 반복.
    만약 모두 1이라면 j는 10이 되어 종료됨. */

    j == 10 ? (valid[1][i] = 1) : (valid[1][i] = 0); /* 만약 j가 10이라면(check가 모두 1이라면)
    valid[1][i]의 값을 1로, 아니라면 0으로 한다. */
}
pthread_exit(NULL);

```

	i	0	1	2	3	4	5	6	7	8
j	0	6	3	9	8	4	1	2	7	5
	1	7	2	4	9	5	3	1	6	8
	2	1	8	5	7	2	6	3	9	4
	3	2	5	6	1	3	7	4	8	9
	4	4	9	1	5	8	2	6	3	7
	5	8	7	3	2	6	9	5	2	1
	6	5	4	4	3	9	8	7	1	6
	7	3	1	8	6	7	5	9	4	2
	8	9	6	7	2	1	4	8	5	3

i = 1
3 2 8 5 9 7 4 1 6

check[]	1	2	3	4	5	6	7	8	9
index	1	1	1	1	1	1	1	1	1

⇒ valid[1][1] = 1

i = 2
9 4 5 6 1 3 4 8 7

check[]	1	2	3	4	5	6	7	8	9
index	1	0	1	1	1	1	1	1	1

⇒ valid[1][2] = 0

변수 j는 sudoku의 행 번호, i는 sudoku의 열을 나타낸다. check[n]은 숫자 n이 배열 안에 있는지 여부를 저장하는 배열이다. sudoku[j][i]의 값(숫자)을 인덱스로 하여 check[sudoku[j][i]]을 1로 할당한다. 만약 check[1]부터 check[9]까지 값이 모두 1 이라면 숫자 1~9가 해당 열에 모두 있다는 뜻이다. 만약 check[1]~check[9] 중 하나라도 1이 아닌 값이 있다면 유효하지 않다.

void *check_subgrid(void *arg)

```

int i, j;

int k = *(int*) arg;

int q = k / 3; /* 해당 서브그리드 인덱스를 3으로 나눈 몫 */
int r = k % 3; /* 해당 서브그리드 인덱스를 3으로 나누고 남은 나머지 */
int check[10] = {}; /* 해당 서브그리드에 1~9까지의 숫자가 있는지 확인하는 배열.
                    숫자 n이 있다면 n번 인덱스의 값은 1, 아니라면 0이 된다. (0번 인덱스는 사용하지 않음) */

for (i = (0 + 3*q); i < (3 + 3*q); i++) {
    for (j = (0 + 3*r); j < (3 + 3*r); j++) {
        check[sudoku[i][j]] = 1; /* sudoku[j][i]의 값(숫자)을 check 배열의 인덱스로 하여 해당 값을 1로 바꾼다. */
    }
}

```

```

for (j = 1; j < 10 && check[j]; j++);          /* check의 1~9번 인덱스의 값이 0일때까지 계속 반복.
                                                  만약 모두 1이라면 j는 10이 되어 종료됨. */
j == 10 ? (valid[2][k] = 1) : (valid[2][k] = 0); /* 만약 j가 10이라면(check가 모두 1이라면)
                                                  valid[2][k]의 값을 1로, 아니면 0으로 한다. */

pthread_exit(NULL);

```

			0			1			2		
			0	1	2	3	4	5	6	7	8
0	0	6	3	9	8	4	1	2	7	5	
	1	7	2	4	9	5	3	1	6	8	
	2	1	8	5	7	2	6	3	9	4	
1	3	2	5	6	1	3	7	4	8	9	
	4	4	9	1	5	8	2	6	3	7	
	5	8	7	3	4	6	9	5	2	1	
2	6	5	4	2	3	9	8	7	1	6	
	7	3	6	8	6	7	5	9	8	2	
	8	9	6	7	2	1	4	8	5	3	

k
 $(k/3, k\%3)$

변수 i는 sudoku의 행 번호, j는 sudoku의 열을 나타낸다. check[n]은 숫자 n이 배열 안에 있는지 여부를 저장하는 배열이다. 서브그리드 인덱스를 k로 받아 i와 j의 범위를 지정하는데 사용한다. 1~9까지 3으로 나눈 몫으로 행과 열 인덱스를 3분류로 나눈다. k를 3으로 나눈 몫을 행, 나머지를 열 인덱스 범위를 지정한다. sudoku[i][j]의 값(숫자)을 인덱스로 하여 check[sudoku[i][j]]을 1로 할당한다. 만약 check[1]부터 check[9]까지 값이 모두 1이라면 숫자 1~9가 해당 서브그리드에 모두 있다는 뜻이다. 만약 check[1]~check[9] 중 하나라도 1이 아닌 값이 있다면 유효하지 않다.

void check_sudoku(void)

```

/*
 * 스레드를 생성하여 각 행을 검사하는 check_rows() 함수를 실행한다.
 */
if ( (&rowtid, NULL, check_rows, NULL) != 0) {
    fprintf(stderr, "pthread_create error: check_rows\n");
    exit(-1);
}

/*
 * 스레드를 생성하여 각 열을 검사하는 check_columns() 함수를 실행한다.
 */
if (pthread_create(&coltid, NULL, check_columns, NULL) != 0) {
    fprintf(stderr, "pthread_create error: check_columns\n");
    exit(-1);
}

/*
 * 9개의 스레드를 생성하여 각 3x3 서브그리드를 검사하는 check_subgrid() 함수를 실행한다.
 * 3x3 서브그리드의 위치를 식별할 수 있는 값을 함수의 인자로 넘긴다.
 */
for (i = 0; i < 9; i++) {
    gid[i] = i;
    if (pthread_create(&subgridtid[i], NULL, check_subgrid, &gid[i]) != 0) {

```

```

        fprintf(stderr, "pthread_create error: check_subgrid\n");
        exit(-1);
    }
}

/*
 * 11개의 스레드가 종료할 때까지 기다린다.
 */
pthread_join(rowtid, NULL);
pthread_join(coltid, NULL);
for (i = 0; i < 9; i++) {
    pthread_join(subgridtid[i], NULL);
}

```

pthread_create 함수를 이용하여 스레드를 생성한다. 첫번째 인자는 thread ID, 세번째 인자는 실행할 함수 포인터, 네번째 인자는 함수에 함께 전달될 인자이다. 스레드가 성공적으로 생성된 경우 0을 반환한다. 각 스레드가 종료될 때까지 기다리는 pthread_join 함수를 사용해 스레드 종료를 기다린다.

2. 컴파일 과정

```

(base) eunbilee@eunbiui-MacBookPro sudoku_#2 % gcc sudoku_skeleton.c -o sudoku
(base) eunbilee@eunbiui-MacBookPro sudoku_#2 % ./sudoku
***** BASIC TEST *****
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
COLS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
---
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 2 6 9 5 2 1
 5 4 4 3 9 8 7 1 6

```

3. 실행 결과물

```

***** BASIC TEST *****
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3

```

```

---
ROWS: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
COLS: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 2 6 9 5 2 1
 5 4 4 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, NO)(6, NO)(7, YES)(8, YES)
COLS: (0, YES)(1, YES)(2, NO)(3, NO)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, NO)(5, YES)(6, NO)(7, YES)(8, YES)
---

```

***** RANDOM TEST *****

```

 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 2 6 2
 1 8 3 9 2 8 3 6 5
 9 4 6 3 9 7 4 3 5
 2 1 5 5 3 2 6 8 2
 6 7 4 4 9 6 4 5 7
 6 2 1 4 5 2 5 2 4
 3 8 5 2 6 9 1 8 6
 7 9 4 7 8 4 4 7 3
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
 6 4 1 5 4 7 6 8 4
 1 7 9 5 1 8 7 4 5
 2 5 8 3 7 2 3 1 6
 5 2 3 4 5 3 9 6 1
 2 4 6 6 9 7 1 8 6
 8 1 7 4 8 9 2 9 1
 8 5 2 3 4 6 6 2 4
 4 1 5 2 6 7 3 1 4
 7 1 6 1 8 1 5 7 2
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
 9 5 2 7 6 1 7 4 2
 3 9 2 7 2 3 8 7 4
 9 1 3 8 6 9 4 9 5
 3 7 9 4 1 2 6 2 7
 4 1 2 8 6 8 6 8 3
 9 3 5 1 2 8 6 4 8
 3 1 5 5 3 9 1 4 3
 1 7 4 4 6 9 6 7 3
 1 2 6 5 2 3 2 9 8
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, NO)
---
 8 9 1 6 5 1 3 9 1
 4 8 2 8 1 4 6 7 2
 8 9 5 9 3 4 7 5 2
 1 2 9 5 3 2 6 9 7
 6 1 4 5 6 9 5 8 6
 1 3 2 3 6 2 8 1 9

```

첫번째 행에 1~9까지의 숫자가 있지만, 결과가 no로 나왔다.
=> 여러 스레드가 자원을 공유하며 동시에 작동하고 있기 때문에 결과가 올바르게 나오지 않는다.

```

3 9 2 7 6 2 4 2 1
8 9 5 9 8 5 2 9 1
6 2 8 2 6 4 4 2 3
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
2 9 4 2 4 8 2 1 6
1 5 8 7 5 4 5 4 7
2 3 8 3 8 6 6 1 2
4 9 1 9 8 4 1 3 8
7 1 8 3 6 5 5 7 6
7 3 5 8 2 4 9 5 4
8 9 3 6 7 5 1 4 8
7 8 9 9 5 3 5 7 9
5 6 4 9 1 4 9 6 4
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, NO)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
6 3 1 6 3 8 7 2 1
9 2 5 5 7 4 3 9 4
8 7 4 2 1 9 8 6 5
5 8 9 6 7 8 7 8 9
7 4 2 2 3 4 3 2 4
3 6 1 9 1 5 6 1 5
8 2 6 3 5 8 3 7 1
3 7 5 7 9 4 8 6 2
9 1 4 1 2 6 4 5 9
---
ROWS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
COLS: (0, NO)(1, NO)(2, NO)(3, NO)(4, NO)(5, NO)(6, NO)(7, NO)(8, NO)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
```

쉬는 과정이 끝난 뒤,
모든 검사 결과가 올바
르다.

4. 문제점과 느낀점

1. 배열 초기화

```

8 1 7 7 8 5 8 3 4
3 5 6 1 4 6 5 1 2
2 9 4 9 2 3 7 6 9
6 9 7 2 5 7 8 4 7
3 2 1 9 4 6 3 1 9
8 4 5 8 1 3 2 5 6
6 3 5 5 4 3 3 4 9
7 4 1 6 7 8 8 6 5
9 8 2 1 2 9 1 7 2
---
ROWS: (0, NO)(1, NO)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
COLS: (0, NO)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
GRID: (0, YES)(1, YES)(2, YES)(3, YES)(4, YES)(5, YES)(6, YES)(7, YES)(8, YES)
---
```

랜덤 테스트가 다 끝난 뒤에도 결과가 올바르지 않게 나왔었다. 해당 문제는 행, 열, 서브그리드 체크마다 배열 초기화가 되지 않아 발생한 문제였다. `int check[10] = {};` 와 같이 배열 초기화를 통해 문제를 해결했다.

2. pthread_exit

```

(base) eunbilee@eunbiui-MacBookPro sudoku_#2 % gcc sudoku.skeleton.c
sudoku.skeleton.c:43:1: warning: non-void function does not return a value [-Wreturn-type]
}
^
sudoku.skeleton.c:65:1: warning: non-void function does not return a value [-Wreturn-type]
}
^
sudoku.skeleton.c:94:1: warning: non-void function does not return a value [-Wreturn-type]
}
^
3 warnings generated.
(base) eunbilee@eunbiui-MacBookPro sudoku_#2 % ./a.out

```

해당 warning 문구는 함수에 return 값이 없을 경우 발생한다. 각각 함수(check_rows, check_columns, check_subgrid) 끝에 `pthread_exit(NULL);` 를 추가한다. `pthread_exit`는 실행중인 스레드를 종료시킨다.

- Thread

이번 과제를 통해 POSIX 스레드(*pthread*)를 사용하며 스레드 생성과 종료 과정을 알아보고 직접 사용할 수 있게 되었다. 스레드는 프로세스 내에서 실행되는 하나의 흐름을 말하며 여러 스레드가 존재할 수 있다. 또한 스레드를 통해 여러 일을 동시에 처리할 수 있다는 것을 알게 되었다. 하지만 하나의 프로세스에서 여러 개의 스레드가 같은 자원과 메모리를 서로 공유하기 때문에 [3]실행결과와 랜덤 테스트 진행 경우 같은 점을 주의해야 한다는 것을 알게 되었다.