

s운영체제론 개인 과제 2 보고서

제출자 : 이찬영 (2019098068, ICT 융합학부)

1. 본인이 작성한 함수에 대한 설명

```
1  /*
2  * Copyright(c) 2021-2023 All rights reserved by Heekuck Oh.
3  * 이 프로그램은 한양대학교 ERICA 컴퓨터학부 학생을 위한 교육용으로 제작되었다.
4  * 한양대학교 ERICA 학생이 아닌 이는 프로그램을 수정하거나 배포할 수 없다.
5  * 프로그램을 수정할 경우 날짜, 학과, 학번, 이름, 수정 내용을 기록한다.
6  * -----한양대학교 ERICA ICT융합학부 2019098068 이찬영-----
7  * 04.01
8  * *check_rows() 함수, *check_columns() 함수, check_subgrid() 함수 안에 내용을 추가하였습니다.
9  * (CODE LINE : 37~57, 66~86, 96~157)
10 * check_sudoku() 함수 내에 pthread_create()와 pthread_join() 부분을 추가하였습니다.
11 * (CODE LINE : 181~208)
12 */
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <stdbool.h>
16 #include <time.h>
17 #include <pthread.h>
18
19 /*
20 * 기본 스도쿠 퍼즐
21 */
22 int sudoku[9][9] = {{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,
23
24 /*
25 * valid[0][0], valid[0][1], ..., valid[0][8]: 각 행이 올바른지 true, 아니면 false
26 * valid[1][0], valid[1][1], ..., valid[1][8]: 각 열이 올바른지 true, 아니면 false
27 * valid[2][0], valid[2][1], ..., valid[2][8]: 각 3x3 그리드가 올바른지 true, 아니면 false
28 */
29 bool valid[3][9];
30
31 /*
32 * 스도쿠 퍼즐의 각 행이 올바른지 검사한다.
33 * 행 번호는 0부터 시작하며, i번 행이 올바른지 valid[0][i]에 true를 기록한다.
34 */
35 void *check_rows(void *arg)
36 {
37     int state; // 행을 검사했을 때 중복되는 상태인지 아닌지를 정하는 변수
38
39     int a; // 임시 변수
40     int temp[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; // sudoku 행을 검사하는 배열
41     for(int i=0; i<9; i++) { // 전체 9개 행을 검사
42         state = 0;
43         a = 0;
44         /*
45          * 스도쿠 행을 순차적으로 돌면서 해당 숫자를 a에 임시 저장하고 temp 배열의 a번째에 1을 넣음
46          */
47         for(int j=0; j<9; j++) {
48             a = sudoku[i][j];
49             temp[a] = 1;
50         }
51         for(int k=1; k<10; k++) {
52             if(temp[k] == 0) { state = 1; } // 행을 검사했을 때, 중복되면(sudoku 규칙에 어긋나면) state = 1
53             temp[k] = 0; // temp 배열을 초기화
54         }
55         if(state == 0) valid[0][i] = true; // state == 0 이면 valid 배열에 true 기록
56         else valid[0][i] = false; // valid 배열에 false
57     }
58     pthread_exit(NULL); // 스레드 종료
59 }
60
61 /*
62 * 스도쿠 퍼즐의 각 열이 올바른지 검사한다.
63 * 열 번호는 0부터 시작하며, j번 열이 올바른지 valid[1][j]에 true를 기록한다.
64 */
65 void *check_columns(void *arg)
66 {
67     int state; // 열을 검사했을 때 중복되는 상태인지 아닌지를 정하는 변수
68     int a; // 임시 변수
69     int temp[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; // sudoku 열을 검사하는 배열
70     for(int i=0; i<9; i++) { // 전체 9개 열을 검사
71         state = 0;
72         a = 0;
73         /*
74          * 스도쿠 열을 순차적으로 돌면서 해당 숫자를 a에 임시 저장하고 temp 배열의 a번째에 1을 넣음
75          */
76     }
```

```

75     for(int j=0; j<9; j++) {
76         a = sudoku[j][i];
77         temp[a] = 1;
78     }
79     for(int k=1; k<10; k++) {
80         if(temp[k] == 0) { state = 1; }    // 열을 검사했을 때, 중복되면(sudoku 규칙에 어긋나면) state = 1
81         temp[k] = 0;                    // temp 배열을 초기화
82     }
83     if(state == 0) valid[1][i] = true;    // state == 0 이면 valid 배열에 true 기록
84     else valid[1][i] = false;            // valid 배열에 false
85 }
86 pthread_exit(NULL);                    // 스레드 종료
87 }
88
89 /*
90 * 스도쿠 퍼즐의 각 3x3 서브그리드가 올바른지 검사한다.
91 * 3x3 서브그리드 번호는 0부터 시작하며, 왼쪽에서 오른쪽으로, 위에서 아래로 증가한다.
92 * k번 서브그리드가 올바르면 valid[2][k]에 true를 기록한다.
93 */
94 void *check_subgrid(void *arg)
95 {
96     int i = *(int *)arg;                // arg 번째 subgrid를 인수로 넘겨 받아 i에 저장함
97     int temp[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};    // subgrid를 검사하는 배열
98     int a;                            // 임시 변수
99     int state;                        // subgrid를 검사했을 때 중복되는 상태인지 아닌지를 정하는 변수
100
101     if (i % 3 == 0) {                // i가 3의 배수일 때 (i = 0, 3, 6)
102
103         for(int j=i; j<=i+2; j++) {    // 해당 subgrid(3x3)를 돌면서 temp 배열의 해당 숫자 위치에 1을 넣는다.
104             for(int k=0; k<=2; k++) {
105                 a = sudoku[j][k];
106                 temp[a] = 1;
107             }
108         }
109         state = 0;
110         for(int k=1; k<10; k++) {        // subgrid 검사
111             if(temp[k] == 0) { state = 1; }    // 스도쿠 조건에 안 맞으면 state = 1

```

```

112         temp[k] = 0;                    // temp 배열 초기화
113     }
114     if(state == 0) valid[2][i] = true;    // 스도쿠 조건에 맞으면 valid 배열에 true
115     else valid[2][i] = false;            // 스도쿠 조건에 안 맞으면 valid 배열에 false
116     state = 0;                          // state = 0으로 초기화
117     a = 0;                              // a = 0으로 초기화
118 }
119
120 else if (i % 3 == 1) {                // i % 3 == 1일 때 (i = 1, 4, 7)
121
122     for(int j=i-1; j<=i+1; j++) {
123         for(int k=3; k<=5; k++) {
124             a = sudoku[j][k];
125             temp[a] = 1;
126         }
127     }
128     state = 0;
129     for(int k=1; k<10; k++) {        // subgrid 검사
130         if(temp[k] == 0) { state = 1; }    // 스도쿠 조건에 안 맞으면 state = 1
131         temp[k] = 0;                    // temp 배열 초기화
132     }
133     if(state == 0) valid[2][i] = true;    // 스도쿠 조건에 맞으면 valid 배열에 true
134     else valid[2][i] = false;            // 스도쿠 조건에 안 맞으면 valid 배열에 false
135     state = 0;                          // state = 0으로 초기화
136     a = 0;                              // a = 0으로 초기화
137 }
138
139 else if (i % 3 == 2) {                // i % 3 == 2일 때 (i = 2, 5, 8)
140
141     for(int j=i-2; j<=i; j++) {
142         for(int k=6; k<=8; k++) {
143             a = sudoku[j][k];
144             temp[a] = 1;
145         }
146     }
147     state = 0;
148     for(int k=1; k<10; k++) {        // subgrid 검사

```

```

148     for(int k=1; k<10; k++) {           // subgrid 검사
149         if(temp[k] == 0) { state = 1; }   // 스도쿠 조건에 안 맞으면 state = 1
150         temp[k] = 0;                     // temp 배열 초기화
151     }
152     if(state == 0) valid[2][i] = true;     // 스도쿠 조건에 맞으면 valid 배열에 true
153     else valid[2][i] = false;             // 스도쿠 조건에 안 맞으면 valid 배열에 false
154     state = 0;                           // state = 0으로 초기화
155     a = 0;                               // a = 0으로 초기화
156 }
157 pthread_exit(NULL);                     // 스레드 종료
158 }
159
160 /*
161  * 스도쿠 퍼즐이 올바르게 구성되어 있는지 11개의 스레드를 생성하여 검증한다.
162  * 한 스레드는 각 행이 올바른지 검사하고, 다른 한 스레드는 각 열이 올바른지 검사한다.
163  * 9개의 3x3 서브그리드에 대한 검증은 9개의 스레드를 생성하여 동시에 검사한다.
164  */
165 void check_sudoku(void)
166 {
167     int i, j;
168     /*
169     * 검증하기 전에 먼저 스도쿠 퍼즐의 값을 출력한다.
170     */
171     for (i = 0; i < 9; ++i) {
172         for (j = 0; j < 9; ++j)
173             printf("%2d", sudoku[i][j]);
174         printf("\n");
175     }
176     printf("---\n");
177     /*
178     * 스레드를 생성하여 각 행을 검사하는 check_rows() 함수를 실행한다.
179     */
180
181     pthread_t r, c;
182     pthread_t tid[9];
183     int arg[9];
184

```

(CODE LINE : 35~58)

*Check_rows() 함수입니다.

이 함수는 9x9 스도쿠에서 9개의 행만 검사하는 기능을 수행합니다.

이중 for문을 통해, 전체 9개의 행을 돌면서, 각 행마다 처음부터 끝까지 9개의 숫자를 탐색하고 그 숫자에 해당하는 index에 temp 배열의 값을 1로 만들어서 중복된 숫자가 각 행마다 있는지 검사할 수 있도록 하였습니다. CODE LINE : 50~53은 temp 배열을 돌면서 값 중 0이 있으면 그 숫자가 행에 없는 것으로 판단을 하고 state = 1로 설정해 줍니다. 동시에 다음 순번에서 temp 배열을 재사용하기 위해 모든 값을 0으로 초기화 합니다. CODE LINE : 54~55은 state = 0이면 정상적으로 검증한 것으로 판단해 valid[0][i] 값을 true로, state = 1이면 비정상적으로 검증한 것으로 판단해 valid[0][i] 값을 false로 설정합니다.

for문을 다 돌고 난 후에는 pthread_exit(NULL)을 통해 스레드를 종료합니다.

(CODE LINE : 64~87)

*Check_column() 함수입니다.

기본적인 작동 원리는 위에 있는 *Check_rows() 함수와 동일합니다.

이 함수는 9x9 스도쿠에서 9개의 열만 검사하는 기능을 수행합니다.

이중 for문을 통해, 전체 9개의 열을 돌면서, 각 열마다 처음부터 끝까지 9개의 숫자를 탐색하고 그 숫자에 해당하는 index에 temp 배열의 값을 1로 만들어서 중복된 숫자가 각 행마다 있는지 검사할 수 있도록 하였습니다. CODE LINE : 79~82은 temp 배열을 돌면서 값 중 0이 있으면 그 숫자가 행에 없는 것으로 판단을 하고 state = 1로 설정해 줍니다. 동시에 다음 순번에서 temp 배열을 재사용하기 위해 모든 값을 0으로 초기화 합니다. CODE LINE : 83~84은 state = 0이면 정상적으로 검증한 것으로 판단해 valid[1][i] 값을 true로, state = 1이면 비정상적으로 검증한 것으로 판단해 valid[1][i] 값을 false로 설정합니다.

for문을 다 돌고 난 후에는 pthread_exit(NULL)을 통해 스레드를 종료합니다.

(CODE LINE : 94~158)

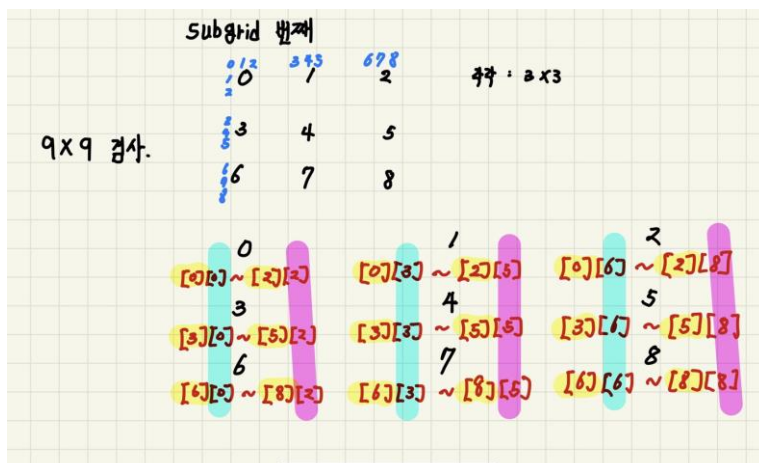
*Check_subgrid() 함수입니다.

이 함수는 pthread_create()를 통해 *arg를 인수로 받아서 (해당 인수는 subgrid의 순번을 나타내는 것으로 설정하였습니다.) i 변수에 대입해 줍니다. 이후에는 i를 3으로 나눈 나머지에 따라서 3개의 case로 분류해 줍니다.

해당하는 순번의 3x3 크기 subgrid를 이중 for문을 통해 9개의 원소를 검사합니다. 해당 원소의 index에 해당하는 temp 값을 1로 만들고 이후에 temp 배열을 처음부터 끝까지 돌면서 0이 있는지를 검사합니다. 0이 있으면 해당 숫자가 subgrid에 없는 것으로 판단하고 state = 1로 만들어줍니다. 동시에 다음 프로세스에서 재사용할 temp 배열의 초기화를 진행합니다.

이후에는 state == 0이면 valid[2][i] 값을 true로 state == 1이면 valid[2][i] 값을 false로 설정해 줍니다. If (i%3 == 0), else if (i % 3 == 1), else if (i % 3 == 2) 안에서 큰 구조는 모두 동일합니다. State와 a값을 초기화 해준 뒤에 pthread_exit(NULL)을 통해 스레드를 종료합니다

<다음과 같은 규칙을 통해 subgrid 함수를 만들었습니다.>



(CODE LINE : 181~183)

check_columns() 함수 실행을 위한 프로세스 변수 r, check_rows() 함수 실행을 위한 프로세스 변수 c, check_subgrid() 함수 실행을 위한 프로세스 배열 tid[9] (9개의 프로세스 실행을 위함)와 정수형 배열 arg[9]를 선언해 줍니다.

```
185  /*
186  * 스레드를 생성하여 각 행을 검사하는 check_rows() 함수를 실행한다.
187  */
188  pthread_create(&r, NULL, check_rows, NULL);
189  /*
190  * 스레드를 생성하여 각 열을 검사하는 check_columns() 함수를 실행한다.
191  */
192  pthread_create(&c, NULL, check_columns, NULL);
193  /*
194  * 9개의 스레드를 생성하여 각 3x3 서브그리드를 검사하는 check_subgrid() 함수를 실행한다.
195  * 3x3 서브그리드의 위치를 식별할 수 있는 값을 함수의 인자로 넘긴다.
196  */
197  for (i = 0; i < 9; i++) {
198      arg[i] = i;
199      pthread_create(tid+i, NULL, check_subgrid, arg+i);
200  }
201  /*
202  * 11개의 스레드가 종료할 때까지 기다린다.
203  */
204  pthread_join(r, NULL);
205  pthread_join(c, NULL);
206  for(i = 0; i < 9; i++) {
207      pthread_join(tid[i], NULL);
208  }
209  /*
210  * 각 행에 대한 검증 결과를 출력한다.
211  */
212  printf("ROWS: ");
213  for (i = 0; i < 9; ++i)
214      printf(valid[0][i] ? "(%d,YES)" : "(%d,NO)", i);
215  printf("\n");
216  /*
217  * 각 열에 대한 검증 결과를 출력한다.
218  */
219  printf("COLS: ");
220  for (i = 0; i < 9; ++i)
221      printf(valid[1][i] ? "(%d,YES)" : "(%d,NO)", i);
```

(CODE LINE : 188~200)

스레드를 하나 생성하여 각 행을 검사하는 check_rows() 함수를 실행합니다.

스레드를 하나 생성하여 각 열을 검사하는 check_columns() 함수를 실행합니다.

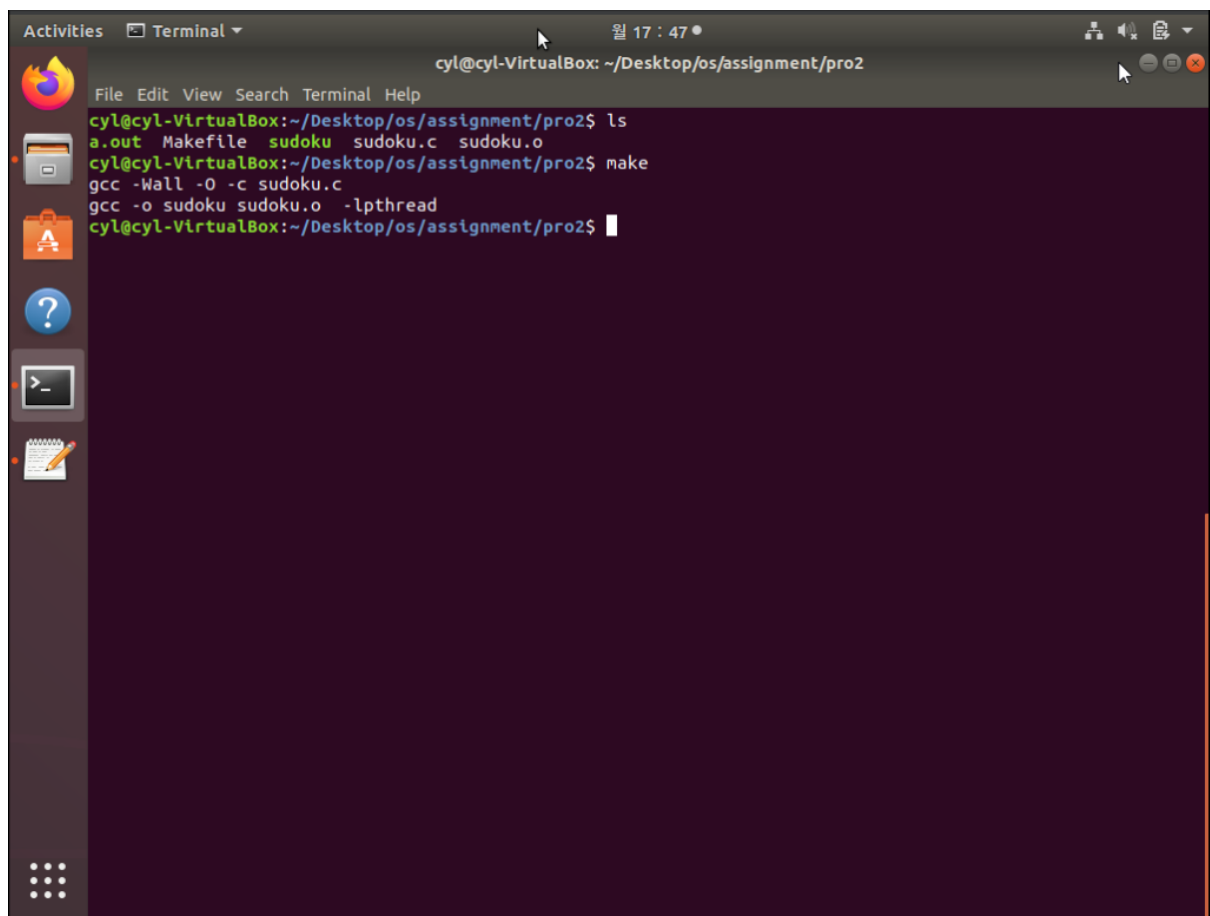
for문을 통해 스레드를 9개 생성하여 각 subgrid을 검사하는 check_subgrid() 함수를 실행합니다.

11개의 스레드가 종료할 때까지 차례대로 기다립니다.

(CODE LINE : 212~)

이후에는 sudoku_skeleton.c 와 동일합니다.

2. 컴파일 과정을 보여주는 화면 캡처

A screenshot of a Linux terminal window titled 'Terminal' with a dark theme. The window shows the user 'cyl' at a 'cyl-VirtualBox' machine in the directory '~/Desktop/os/assignment/pro2'. The user has run 'ls' showing files 'a.out', 'Makefile', 'sudoku', 'sudoku.c', and 'sudoku.o'. Then they ran 'make', which executed 'gcc -Wall -O -c sudoku.c' followed by 'gcc -o sudoku sudoku.o -lpthread'. The terminal output is as follows:

```
cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$ ls
a.out  Makefile  sudoku  sudoku.c  sudoku.o
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$ make
gcc -Wall -O -c sudoku.c
gcc -o sudoku sudoku.o -lpthread
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$
```

교수님께서 주신 Makefile로 make 명령어를 통해 sudoku.c 파일을 컴파일한 화면을 캡처하였습니다. 아무 오류 없이 컴파일이 잘 수행된 것을 확인할 수 있습니다.

3. 실행 결과물에 대한 상세한 설명

```
Activities Terminal 월 17 : 47
cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
File Edit View Search Terminal Help
gcc -Wall -O -c sudoku.c
gcc -o sudoku sudoku.o -lpthread
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$ ./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)
---
 2 3 9 8 4 1 2 7 5
 7 6 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 4 6 3 7
 8 7 3 2 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 3 2
 9 6 7 2 1 4 8 5 4
---
ROWS: (0,NO)(1,NO)(2,YES)(3,YES)(4,NO)(5,NO)(6,YES)(7,NO)(8,NO)
COLS: (0,NO)(1,NO)(2,YES)(3,NO)(4,YES)(5,NO)(6,YES)(7,NO)(8,NO)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
---
***** RANDOM 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
```

```
Activities Terminal 월 17 : 17 cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
File Edit View Search Terminal Help
***** RANDOM 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,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 2 7 2 4 7 8 6 1
7 5 3 5 9 3 5 6 2
5 4 6 4 7 3 1 9 6
7 5 8 3 4 2 4 8 5
9 4 6 8 2 1 6 1 5
6 4 7 9 7 3 5 3 5
3 1 2 8 7 4 2 5 8
2 8 9 1 4 5 6 1 3
8 3 2 8 5 7 8 5 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)
---
6 4 3 9 8 1 8 9 7
7 4 1 6 9 5 8 3 7
6 7 2 6 2 4 5 8 1
8 7 1 1 4 7 1 8 4
4 5 9 1 9 3 4 5 9
5 4 8 6 7 5 3 1 2
6 7 4 2 5 8 2 4 9
8 2 7 4 2 7 5 6 2
5 1 8 3 4 3 4 6 1
---
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)
---
8 2 4 9 6 3 7 9 2
4 8 6 4 2 3 9 1 4
1 3 5 5 9 1 9 7 2
5 2 8 4 5 2 7 2 8
4 6 1 6 1 9 6 8 9
8 7 3 3 5 6 8 3 7
3 8 5 4 1 8 8 1 3
4 2 9 9 5 7 5 8 9
7 1 9 1 8 6 1 6 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)
---
4 9 4 4 1 4 8 1 6
3 6 9 8 2 7 4 6 1
4 3 6 3 9 2 1 2 8
6 7 9 1 2 4 3 1 4
2 7 8 9 5 6 7 8 9
4 5 7 9 5 3 7 6 3
5 8 4 2 6 4 4 6 9
8 1 5 3 1 9 2 6 7
8 2 6 7 8 5 8 3 4
```

```
Activities Terminal 월 17 : 47 cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
File Edit View Search Terminal Help
---
6 4 3 9 8 1 8 9 7
7 4 1 6 9 5 8 3 7
6 7 2 6 2 4 5 8 1
8 7 1 1 4 7 1 8 4
4 5 9 1 9 3 4 5 9
5 4 8 6 7 5 3 1 2
6 7 4 2 5 8 2 4 9
8 2 7 4 2 7 5 6 2
5 1 8 3 4 3 4 6 1
---
ROWS: (0,NO)(1,YES)(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)
---
8 2 4 9 6 3 7 9 2
4 8 6 4 2 3 9 1 4
1 3 5 5 9 1 9 7 2
5 2 8 4 5 2 7 2 8
4 6 1 6 1 9 6 8 9
8 7 3 3 5 6 8 3 7
3 8 5 4 1 8 8 1 3
4 2 9 9 5 7 5 8 9
7 1 9 1 8 6 1 6 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)
---
4 9 4 4 1 4 8 1 6
3 6 9 8 2 7 4 6 1
4 3 6 3 9 2 1 2 8
6 7 9 1 2 4 3 1 4
2 7 8 9 5 6 7 8 9
4 5 7 9 5 3 7 6 3
5 8 4 2 6 4 4 6 9
8 1 5 3 1 9 2 6 7
8 2 6 7 8 5 8 3 4
```



```
Activities Terminal 월 17 : 48
cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
File Edit View Search Terminal Help
8 7 3 3 5 6 8 3 7
3 8 5 4 1 8 8 1 3
4 2 9 9 5 7 5 8 9
7 1 9 1 8 6 1 6 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)
---
4 9 4 4 1 4 8 1 6
3 6 9 8 2 7 4 6 1
4 3 6 3 9 2 1 2 8
6 7 9 1 2 4 3 1 4
2 7 8 9 5 6 7 8 9
4 5 7 9 5 3 7 6 3
5 8 4 2 6 4 4 6 9
8 1 5 3 1 9 2 6 7
8 2 6 7 8 5 8 3 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,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
---
4 9 6 4 9 7 7 5 4
8 7 3 8 1 2 9 6 8
5 1 2 5 3 6 2 1 3
8 5 9 9 7 3 4 3 6
2 7 4 4 5 2 9 5 1
1 3 6 1 8 6 2 7 8
4 1 6 5 1 7 9 2 4
8 7 9 8 4 2 3 5 7
5 2 3 6 3 9 8 1 6
---
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)
---
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$
```

Basic test 에서는 스도쿠 행 검사와 열 검사, subgrid 검사를 정상적으로 잘 수행함을 알 수 있습니다. 반면, Random test 에서는 정상적인 검증 작업이 수행되지 않습니다. 이는 RANDOM TEST에서 무작위로 스도쿠 퍼즐을 섞는 중에 프로세스가 종료되지 않은 상태에서 check_sudoku() 함수를 수행하기 때문이며, 스도쿠 퍼즐을 섞는 와중에 출력되므로 정상적인 스도쿠 형태가 아닙니다. 또한 공유자원에 대한 동기화가 이루어지지 않으므로 스레드들이 동시에 해당 자원에 접근할 때, 충돌이 발생합니다. 이러한 경쟁 상태 때문에 결론적으로 검증이 정상적으로 수행되지 않는 결과를 초래합니다.

```
Activities Terminal 수 20 : 45
cyl@cyl-VirtualBox: ~/Desktop/os/assignment/pro2
File Edit View Search Terminal Help
2 6 8 9 1 6 2 8 5
4 3 1 5 3 8 9 3 1
9 5 7 2 7 4 4 6 7
---
ROWS: (0,NO)(1,NO)(2,NO)(3,YES)(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)
---
5 4 2 8 6 1 4 8 3
4 6 5 4 3 9 4 9 6
7 3 1 6 5 1 1 5 2
7 4 9 8 1 6 1 7 3
5 6 3 4 9 3 2 6 8
8 2 1 7 2 5 5 4 9
8 1 9 6 1 9 8 1 2
2 4 5 4 8 3 4 9 6
6 3 7 2 7 5 7 5 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)
---
9 6 4 8 1 9 9 1 2
8 1 5 6 5 7 7 5 8
3 2 7 4 2 3 3 4 6
7 8 9 6 5 1 9 4 6
1 3 2 8 3 4 5 2 7
6 4 5 7 2 9 1 3 8
3 1 4 1 4 7 6 3 9
5 9 6 5 6 2 2 8 5
2 8 7 3 8 9 7 4 1
---
ROWS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,YES)(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)
---
빙고! 5번 행이 맞았습니다.
cyl@cyl-VirtualBox:~/Desktop/os/assignment/pro2$
```

→ 하지만 ./sudoku를 여러 번 실행해보면 위와 같이 빙고가 뜨는 경우도 간혹 있었습니다. 이 경우는 프로세스가 종료된 뒤에 check_sudoku()를 한 결과 우연히 일치해서 값이 출력되는 것으로 추정됩니다.

4. 과제를 수행하면서 경험한 문제점과 느낀 점

이번 과제를 통해 수업시간 때 다룬 스레드 개념에 대해서 깊이 이해를 할 수 있었습니다. 스레드 생성과 종료에 대해서는 금요일 실습시간 때 실습을 해 보았기 때문에 과제를 수행하는 중에 큰 어려움은 없었습니다. 다만, 과제를 위해 코드 전체를 이해하는 과정에서 다중 스레드가 어떻게 동작하는지, 스레드 실행 중에 충돌은 어떻게 일어나는지에 대한 과정들을 세세하게 이해하고 관찰할 수 있었습니다.

또한 프로세스는 운영체제로부터 자원을 할당 받아 실행되는 프로그램의 인스턴스인 만큼 독립된 메모리 공간과 시스템 자원을 가지며, 각 프로세스는 서로 완전히 독립적으로 실행되는 데에 반해 스레드는 하나의 프로세스 내에서 실행되는 작은 실행 단위이고, 스레드는 프로세스 내의 자원을 공유하며, 각 스레드는 독립적으로 실행될 수 없다는 차이를 알 수 있었습니다.

이러한 차이점 때문에 프로세스 간 통신이 스레드 간 통신보다 더 복잡하고 느리며, 스레드를 사용하면 프로그램의 처리 속도를 높일 수 있다는 것도 추가적으로 알게 되었습니다.

스도쿠 함수를 판별하는 부분을 코딩할 때는 행과 열 검증 부분은 서로 겹치는 것이 많아 최대한 비슷하게 하려고 노력하였고, subgrid를 판별하는 부분을 코딩할 때는 크게 나눌 수 있는 case들을 먼저 분류하고 subgrid 부분에서는 서로의 공통점을 찾으려고 노력한 것이 많은 도움이 되었습니다.

운영체제 수업에서 앞으로도 남은 기간 성실히 임하겠습니다. 감사합니다.