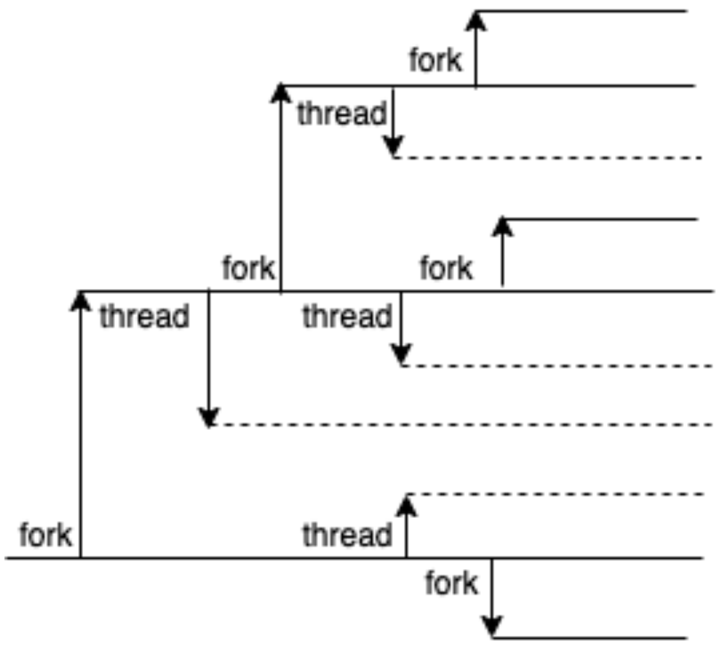
Le Duy Vu Homework 2 CS 149

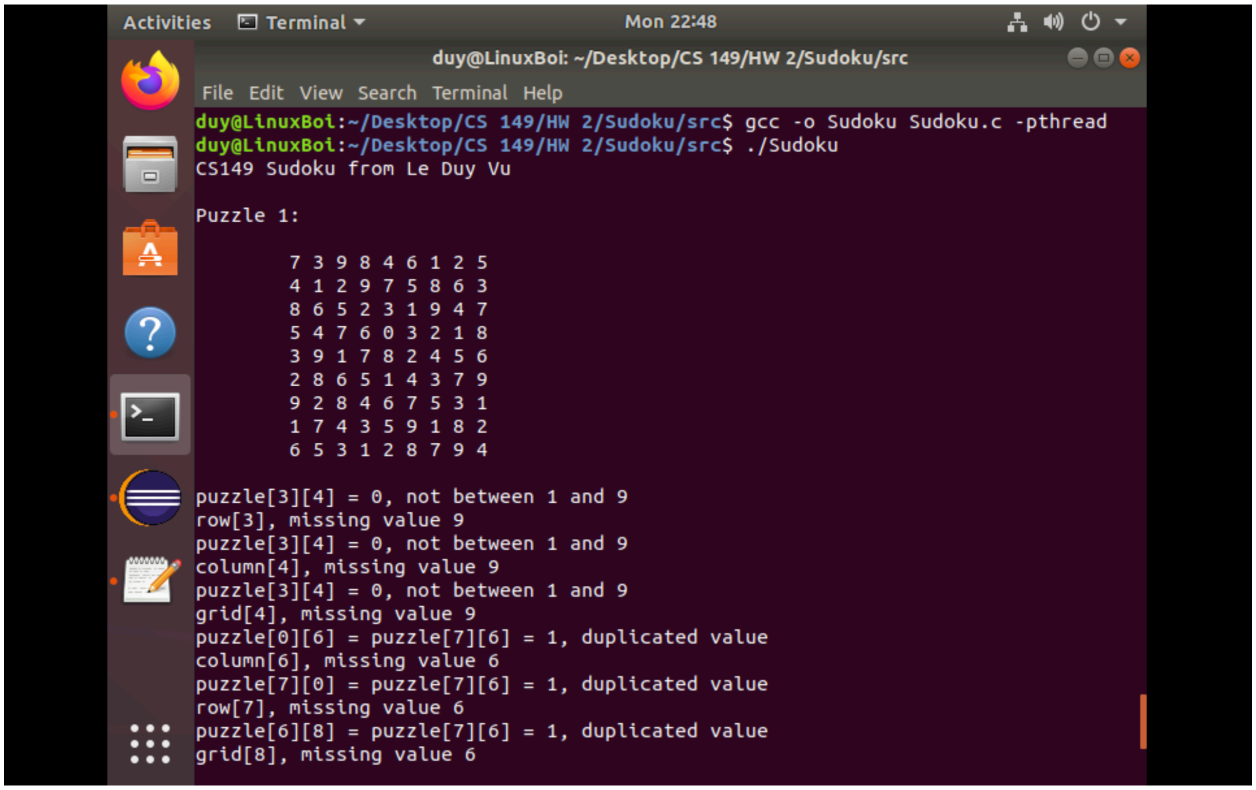
1.
   a. Total number of unique processes created (including root process): 6
   b. Total number of unique threads created by pthread_create(): 4



2.
   a. Puzzle 2 is valid so it has no error.
      Puzzle 1 has several errors:
   - In grid 4, row 3 and column 4, unique pair of errors is {value 0 not between 1 and 9, missing value 9}.
   - In grid 8, row 7 and column 6, unique pair of errors is {duplicated value 1, missing value 6}.
   - These are 2 pairs of errors happening in puzzle 1, there can be a different situation where all 3 types of error happen, such as duplicated 0. Then all 3 errors duplicated value, value not between 1 and 9, and missing value will happen.

   b.

Puzzle	Valid?	Total number of errors	Number of errors for each error type
puzzle1	N	12	Duplicated value: 3, not between 1 and 9: 3, missing value: 6
puzzle2	Y	0	0


Activities
Terminal
Mon 22:48
duy@LinuxBoi: ~/Desktop/CS 149/HW 2/Sudoku/src
File Edit View Search Terminal Help

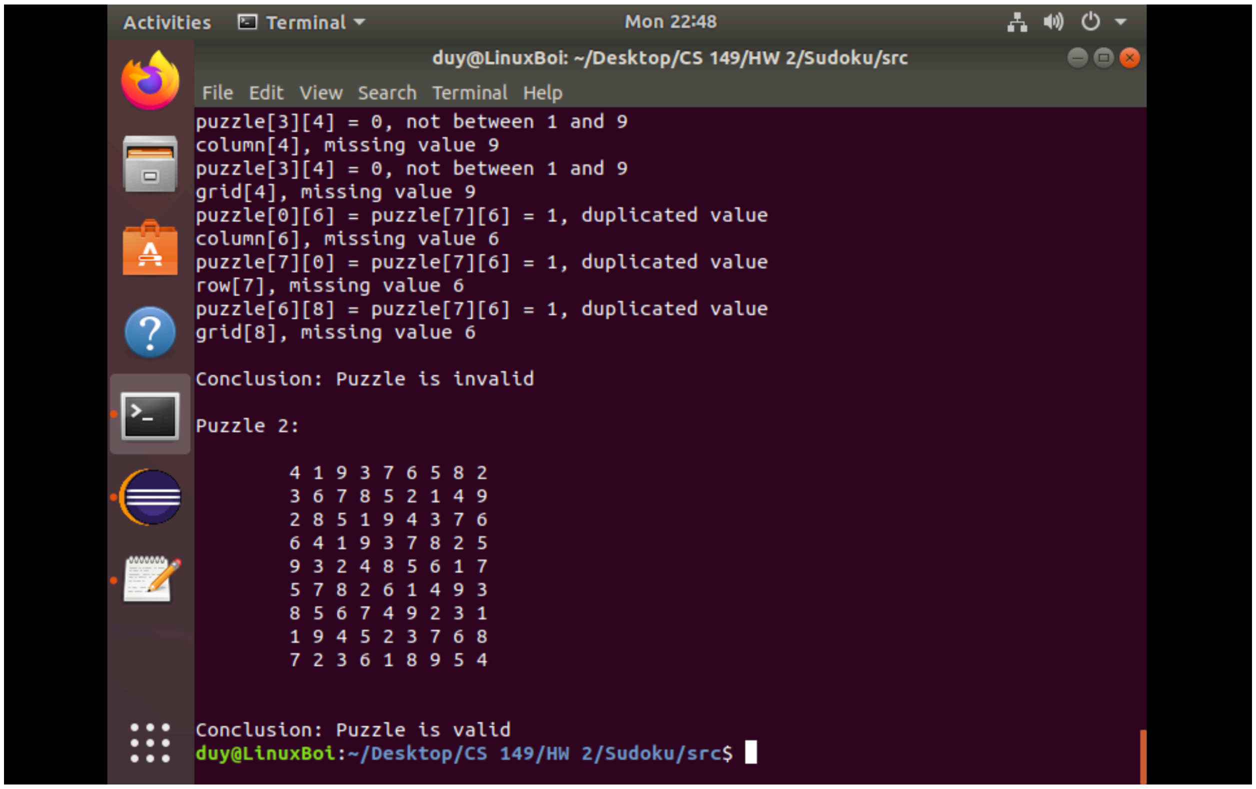
```
duy@LinuxBoi:~/Desktop/CS 149/HW 2/Sudoku/src$ gcc -o Sudoku Sudoku.c -pthread
duy@LinuxBoi:~/Desktop/CS 149/HW 2/Sudoku/src$ ./Sudoku
CS149 Sudoku from Le Duy Vu

Puzzle 1:

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

puzzle[3][4] = 0, not between 1 and 9
row[3], missing value 9
puzzle[3][4] = 0, not between 1 and 9
column[4], missing value 9
puzzle[3][4] = 0, not between 1 and 9
grid[4], missing value 9
puzzle[0][6] = puzzle[7][6] = 1, duplicated value
column[6], missing value 6
puzzle[7][0] = puzzle[7][6] = 1, duplicated value
row[7], missing value 6
puzzle[6][8] = puzzle[7][6] = 1, duplicated value
grid[8], missing value 6
```

Screenshot 1


Activities
Terminal
Mon 22:48
duy@LinuxBoi: ~/Desktop/CS 149/HW 2/Sudoku/src
File Edit View Search Terminal Help

```
puzzle[3][4] = 0, not between 1 and 9
column[4], missing value 9
puzzle[3][4] = 0, not between 1 and 9
grid[4], missing value 9
puzzle[0][6] = puzzle[7][6] = 1, duplicated value
column[6], missing value 6
puzzle[7][0] = puzzle[7][6] = 1, duplicated value
row[7], missing value 6
puzzle[6][8] = puzzle[7][6] = 1, duplicated value
grid[8], missing value 6

Conclusion: Puzzle is invalid

Puzzle 2:

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


Conclusion: Puzzle is valid
duy@LinuxBoi:~/Desktop/CS 149/HW 2/Sudoku/src$
```

Screenshot 2

c. I check all 3 types of error in 1 function:

```
int j, valid = 1 ;
int missing[GRID_SIZE] = {0} ;

for (int i = 0; i < GRID_SIZE; i++)
{
   if (*(param->data + i) < 1 || *(param->data + i) > 9)
   {
      printf("puzzle[%d][%d] = %d, not between 1 and 9\n”, param->index, i, *(param->data +i));
      valid = 0 ;
   }

   for (j = i + 1; j < GRID_SIZE; j++)
      if (*(param->data + i) == *(param->data + j))
      {
         printf("puzzle[%d][%d] = puzzle[%d][%d] = %d, duplicated value\n", param->index, i,
                  param->index, j, *(param->data + i)) ;
         valid = 0 ;
      }
   for (j = 0; j < GRID_SIZE; j++)
      if (*(param->data + i) == j + 1)
         missing[j] = 1 ;
}

for (j = 0; j < GRID_SIZE; j++)
   if (missing[j] == 0)
   {
      printf("row[%d], missing value %d\n", param->index, j + 1) ;
      valid = 0 ;
   }
```

First, I run a big for loop for each integer in the array. First thing I do in the loop is check each int if it lies outside the legal range from 1 to 9 (**check range 1 to 9**). Next, I use brute force to do another loop inside the outer loop to check for duplicate. Each duplicated value found will be printed out (**check for duplicate**). Finally, I use an array of length 9 together with another small loop inside outer loop to check for appearance of each int from 1 to 9. After the outer loop, I check for the appearance array to find which value is missing (**check for missing value**).

d.
```
int flags[THREADS_NUM] ; //global variable
…
//Inside worker thread function
flags[j] = valid ;
pthread_exit(flags + j) ;
…
//Inside main
for (int i = 0; i < THREADS_NUM; i++)
   if (flags[j] == 0)
      valid = 0 ;
if (valid == 0)
   puts("\nConclusion: Puzzle is invalid") ;
else
   puts("\nConclusion: Puzzle is valid") ;
```

In this assignment, I declared a global array of int containing valid flags for the puzzle. 0 means invalid, 1 means valid. Worker threads deposit its result to the global variable, main thread uses it to access the validity of the puzzle. If the array contains all 1, the puzzle is valid. But if there's at least a 0 in there, it is not valid.