```
/*
================================================================================
 *       Filename:  sudoku.c     Version:  1.0
 *        Created:  2015/09/29   Author:  Shuaiqi Cao
 * ==============================================================================
 */
#include <pthread.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

void *grid_checker(void *parameters);
void *row_checker(void *parameters);
void *col_checker(void *parameters);

int sudoku[9][9];
bool sudoku_valid=true;
typedef struct parameters{
        int row;
        int col;
}parameters;

#define DELIM " \t\r\n\a"
#define READIN_BUF_SIZE 300
int main(int argc, char *argv[])
{
        /*Read the input file "sudoku.txt" as a file stream,
        then convert the stream to a string*/
        FILE *stream;
        stream = fopen("sudoku.txt","r");
        char input[READIN_BUF_SIZE];
        fgets(input,READIN_BUF_SIZE,stream);

        /*Use strtok() to split the input string, convert the
        char in string to integer, and store in the 9*9 array suduko[][]*/
        char *number;
        int row=0;
        int col=0;
        for(row; row<9; row++){
                for(col; col<9; col++){
                        number = strtok(input, DELIM);
                while (number != NULL){
                        sudoku[row][col] = atoi(number);
                        col++;
                        number = strtok(NULL, DELIM);
                }
```

```
		}
	}

	/*Create 27 parameters structs, the 1st to the 9th will represent the start
	position of subgrid checking; the 10th to the 18th will represent the start
	position of row checking; the 19th to the 27th will represent the start
	position of column checking. Assign the position*/
	parameters position[27];
	int i, j;
	int n=0;
	for(i=0; i<3; i++){
		for(j=0; j<3; j++){
			position[n].row = 0+3*i;
			position[n].col = 0+3*j;
			n++;
		}
	}
	for(n=0;n<9;n++){
		position[n+9].row=n;
		position[n+9].col=0;
	}
	for(n=0;n<9;n++){
		position[n+18].row=0;
		position[n+18].col=n;
	}

	/*Create 27 threads. 1th to 9th for subgrid checking; 10th to the 18th for
	row checking; 19th to the 27th for column checking. All of the threads are
	created and then perform the pthread_join() to wait the thread terminate*/
	pthread_t tid[27];
	pthread_attr_t attr[27];
	int grid_number;
	int row_number;
	int col_number;

	for(grid_number=0; grid_number<9; grid_number++){
		pthread_attr_init(&attr[grid_number]);
		pthread_create(&tid[grid_number],&attr[grid_number],grid_checker,
(void*)&position[grid_number]);
	}

	for(row_number=0; row_number<9; row_number++){
		pthread_attr_init(&attr[row_number+9]);
		pthread_create(&tid[row_number+9],&attr[row_number+9],row_checker,
(void*)&position[row_number+9]);
	}

	for(col_number=0; col_number<9; col_number++){
		pthread_attr_init(&attr[col_number+18]);
```

```
                pthread_create(&tid[col_number+18],&attr[col_number+18],col_checker,
(void*)&position[col_number+18]);
        }

        int k;
        for(k=0;k<27;k++){
                pthread_join(tid[k],NULL);
        }
        /*Check the global bool variable sudoku_valid. This variable will be changed
        to false if any digit is missed in any subgrid, row or column. So, if it still
        remains true, output that the sudoku is valid*/
        if(sudoku_valid){
                printf("This is a valid Sudoku.\n");
        }else;

        return 0;
}

void *grid_checker(void *parameters)
{
        /*Set a counter array to count the number of 1-9. Use loop to scan the area
         needed to be checked. If it contains all 1-9, the counter array will be all 1*/
        struct parameters* p=(struct parameters*)parameters;
        int row=p->row;
        int col=p->col;
        int counter[9]={0,0,0,0,0,0,0,0,0};
        int row_end=row+3;
        int col_end=col+3;
        int i,j;
        for(i=row; i<row_end; i++){
                for(j=col; j<col_end; j++){
                        if(sudoku[i][j]==1){
                                counter[0]++;
                        }else if(sudoku[i][j]==2){
                                counter[1]++;
                        }else if(sudoku[i][j]==3){
                                counter[2]++;
                        }else if(sudoku[i][j]==4){
                                counter[3]++;
                        }else if(sudoku[i][j]==5){
                                counter[4]++;
                        }else if(sudoku[i][j]==6){
                                counter[5]++;
                        }else if(sudoku[i][j]==7){
                                counter[6]++;
                        }else if(sudoku[i][j]==8){
                                counter[7]++;
                        }else if(sudoku[i][j]==9){
                                counter[8]++;
```

```
                        }else;

                }
        }

        /*Check wether this area contain all 1-9, if not, put all the missing digits
        into the missing_digit[] array*/
        int
valid=counter[0]&&counter[1]&&counter[2]&&counter[3]&&counter[4]&&counter[5]&&counter[6]&
&counter[7]&&counter[8];
        int missing_digit[9];
        int a;
        int b=0;
        for(a=0; a<9; a++){
                if(counter[a]==0){
                        missing_digit[b]=a+1;
                        b++;
                }
        }

        /*If this area is valid, remain the global variable sudoku_valid true; if not,
        print out the information of this area missing digits, and change the
        sudoku_valid to false*/
        if(valid){
                //sudoku_valid=true;
        }else{
                sudoku_valid=false;
                printf("Subgrid: row %d to %d and col %d to %d; miss:",row+1, row+3, col+1, col
+3);
                int t;
                for(t=0;t<b;t++){
                        printf("%d ",missing_digit[t]);
                }
                printf("\n");
        }

}

/*Following row_checker() and col_checker() use same strategy as the grid_checker()*/
void *row_checker(void *parameters)
{
        struct parameters* p=(struct parameters*)parameters;
        int row=p->row;
        int col=p->col;
        int counter[9]={0,0,0,0,0,0,0,0,0};
        int i;
        for(i=0; i<9; i++){
                if(sudoku[row][i]==1){
                        counter[0]++;
```

```
                }else if(sudoku[row][i]==2){
                        counter[1]++;
                }else if(sudoku[row][i]==3){
                        counter[2]++;
                }else if(sudoku[row][i]==4){
                        counter[3]++;
                }else if(sudoku[row][i]==5){
                        counter[4]++;
                }else if(sudoku[row][i]==6){
                        counter[5]++;
                }else if(sudoku[row][i]==7){
                        counter[6]++;
                }else if(sudoku[row][i]==8){
                        counter[7]++;
                }else if(sudoku[row][i]==9){
                        counter[8]++;
                }else;
        }

        int
valid=counter[0]&&counter[1]&&counter[2]&&counter[3]&&counter[4]&&counter[5]&&counter[6]&
&counter[7]&&counter[8];
        int missing_digit[9];
        int a;
        int b=0;
        for(a=0; a<9; a++){
                if(counter[a]==0){
                        missing_digit[b]=a+1;
                        b++;
                }
        }

        if(valid){
                //sudoku_valid=true;
        }else{
                sudoku_valid=false;
                printf("Row: %d; miss:",row+1);
                int t;
                for(t=0;t<b;t++){
                        printf("%d ",missing_digit[t]);
                }
                printf("\n");
        }
}

void *col_checker(void *parameters)
{
        struct parameters* p=(struct parameters*)parameters;
        int row=p->row;
```

```
	int col=p->col;
	int counter[9]={0,0,0,0,0,0,0,0,0};
	int i;
	for(i=0; i<9; i++){
		if(sudoku[i][col]==1){
			counter[0]++;
		}else if(sudoku[i][col]==2){
			counter[1]++;
		}else if(sudoku[i][col]==3){
			counter[2]++;
		}else if(sudoku[i][col]==4){
			counter[3]++;
		}else if(sudoku[i][col]==5){
			counter[4]++;
		}else if(sudoku[i][col]==6){
			counter[5]++;
		}else if(sudoku[i][col]==7){
			counter[6]++;
		}else if(sudoku[i][col]==8){
			counter[7]++;
		}else if(sudoku[i][col]==9){
			counter[8]++;
		}else;
	}

	int
valid=counter[0]&&counter[1]&&counter[2]&&counter[3]&&counter[4]&&counter[5]&&counter[6]&
&counter[7]&&counter[8];
	int missing_digit[9];
	int a;
	int b=0;
	for(a=0; a<9; a++){
		if(counter[a]==0){
			missing_digit[b]=a+1;
			b++;
		}
	}

	if(valid){
	}else{
		sudoku_valid=false;
		printf("Column: %d; miss:",col+1);
		int t;
		for(t=0;t<b;t++){
			printf("%d ",missing_digit[t]);
		}
		printf("\n");
	}
}
```