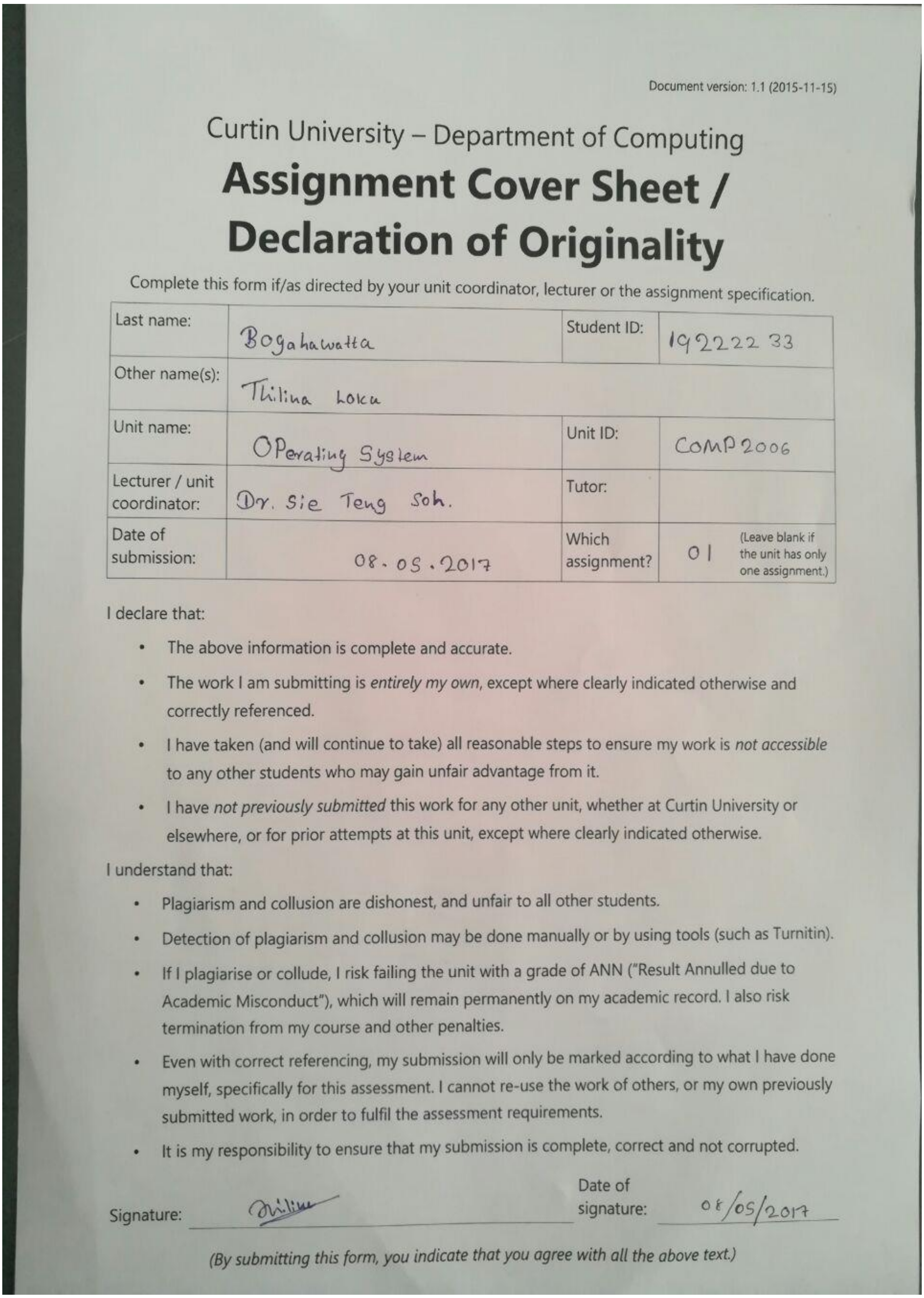


Document version: 1.1 (2015-11-15)

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:	Bogahawatta	Student ID:	19222233
Other name(s):	Thilina Loku		
Unit name:	OPerating System	Unit ID:	COMP2006
Lecturer / unit coordinator:	Dr. Sie Teng Soh.	Tutor:	
Date of submission:	08.05.2017	Which assignment?	01 (Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: [illegible] Date of signature: 08/05/2017

(By submitting this form, you indicate that you agree with all the above text.)

Curtin University

Multitasked Sudoku Solution Validator

Operating Systems

Assignment 2017

Submitted by
Thilina Loku Bogahawatta

CURTIN ID : 19222233
SLIIT ID : IT15005786

Table of content

1. Introduction

We need to implement a program by using C programming language for Multitasked Sudoku Solution Validator. As well as this program uses command line arguments to get the inputs for the program. There are two parts. So we implement same validation programme by two different ways, by using processes and threads.

2. Source code

2.1 Processes

```
        #include<stdlib.h>
#include<stdio.h>               /* exit(), malloc(), free() */
#include<sys/types.h>           /* key_t, sem_t, pid_t */
#include<sys/shm.h>             /* shmat(), IPC_RMID*/
#include<semaphore.h>           /* sem_open(), sem_destroy(),
sem_wait()*/
#include<fcntl.h>               /* O_CREAT, O_EXEC */


typedef struct structure{
int pid;

}subt;

int M,N;
int *Buffer1;
int segid1,y=1,number1;
int i,j,l;
FILE *file1;
int c,d,forvalue;
sem_t *sem;
unsigned int semValue;
int *temp;
int results[9]={0};
int results2[9]={0};


int main(int argc,char *argv[]){
```

```
subt *sub;

/*Getting command line arguments*/

M=atoi(argv[2]);
N=atoi(argv[3]);

/*Creating and attaching Buffer1, shared memory */

if((segid1=shmget(2000000000,sizeof(int)*M*N,IPC_CREAT|06
66))<0)
{
perror("shmget1");
exit(1);
}

if((Buffer1=(int *)shmat(segid1,NULL,0))<0){
perror("shmat");
exit(1);

}


semValue=1;
/*Semphore value, if this value is changed to 0
 * it dosen't  access to cririical section
/* initialize semaphores for shared processes */
sem=sem_open("PSEM",O_CREAT|O_EXCL,0644,semValue);
/* name of semaphore is "pSem", semaphore is reached using
this name */
sem_unlink("PSEM");/* unlink prevents the semaphore existing
forever
/* if a crash occurs during the execution
```

```
/*Opening files */
file1=fopen(argv[1],"r");

if(file1==NULL ){

         printf("File open Failed!!");
         exit(1);

}

for(i=0;i<M;i++){
         for(j=0;j<N;j++){
          if(fscanf(file1,"%d",&number1)!=EOF){
                  Buffer1[i*N+j]=number1;
          }
         }
}



printf("    TABLE  \n");

for(i=0;i<M;i++){
         for(j=0;j<N;j++){

          printf("  %d",Buffer1[i*N+j]);

         }
printf("\n");
}
```

```
  for(i=0;i<=M;i++){


        sub->pid=fork();

        if(i==9){

                if(sub->pid==0){
                printf("Validation_Result_from_Process
(%d):%d\n",i+1,getpid());
                printf("Columns %d of %d Valid  \n",i,i);

                // columns cheak
                for(j=0;j<N;j++){


                        int tmp;

                        int array_value= Buffer1[i*N+j];

                        if(results2[0]==0){

                                results2[0]=array_value;




                        }else
                        {
                        }

                        int n=sizeof(results2);

                        for(l=1;l<=8;l++){



                        if(results2[l]==array_value){

                                printf("Invalid Columns: %d",results2[j]);

                        }else{

                                results2[l]=array_value;


                        }

                        }



                        }

        }

                }


                }
        else{
        int sum2=0;
        if(sub->pid<0){

        printf("Error");
        }

        else if(sub->pid==0)
                {
                sem_wait(sem);

printf("Validation_Result_from_Process
(%d):%d\n",i+1,getpid());


                //row cheaking
                for(j=0;j<N;j++){


                        int tmp;

                        int array_value= Buffer1[i*N+j];

                        if(results[0]==0){

                                results[0]=array_value;




                        }else
                        {
```

```
                    int n=sizeof(results);

                    for(l=1;l<=8;l++){



                    if(results[l]==array_value){

                        printf("Invalid Row: %d",results[j]);

                    }else{

                        results[l]=array_value;


                    }

                    }


                    }

            }
        printf("Row %d is Valid \n",i+1);



                printf("\n");
                sem_post(sem);
                exit(0);
```

```
                    }

                else
                {
                wait(NULL);
                }
        }

        }
        sem_destroy(sem);
        }
```

2.2 Threads

```
#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>


int number1,i,j,M,N,K;

int A[9][9]; //Initializing arrays by setting default values

FILE *file1;

/* The mutex lock */


pthread_mutex_t mutex;

pthread_t tidp,tidc[9];      //Thread ID

pthread_attr_t attr; //Set of thread attributes
```

```
void *producer(void *param); /* the producer thread */
void *consumer(void *param); /* the consumer thread */

/* Producer Thread */

void *producer(void *param ) {



pthread_mutex_lock(&mutex); //acquire the mutex lock

//Get data from the textfiles

for(i=0;i<M;i++){
        for(j=0;j<N;j++){
                if(fscanf(file1,"%d",&number1)!=EOF){
                A[i][j]=number1;
        }
        }
}



pthread_mutex_unlock(&mutex); //release the mutex lock
pthread_exit(0);
}

void *consumer(void *param) {

pthread_mutex_lock(&mutex); //acquire the mutex lock

int va=(int)param;

for(j=0;j<K;j++){

sum2=sum2+A[va][j];
}
printf("\nValidation_Result_from_Thread (%d):%d\n",va+1,tidc[va]);




pthread_mutex_unlock(&mutex); //release the mutex lock
pthread_exit(0);
}
```

```
int main(int argc,char *argv[] ){


file1=fopen(argv[1],"r");
M=atoi(argv[2]);
N=atoi(argv[3]);



A[M][N];


/* Create the mutex lock */
pthread_mutex_init(&mutex,NULL);
/*Create the thread*/
pthread_create(&tidp,NULL,producer,NULL);


pthread_join(tidp,NULL);


printf("   TABLE  \n");
for(i=0;i<M;i++){
        for(j=0;j<N;j++){
          printf("%d ",A[i][j]);
        }
printf("\n");
}
int r=0;
/*Create the thread*/
for(r=0;r<M;r++){
        pthread_create(&tidc[r],NULL,consumer,(void*)r);
}
for(r=0;r<M;r++){
        pthread_join(tidc[r],NULL);
}


pthread_mutex_destroy(&mutex);
pthread_exit(0);


}
```

3. How to compile

3.1 Processes

To compile **gcc –opro OS_Proccess.c –lpthread**

Then **./pro data1.txt 9 9**

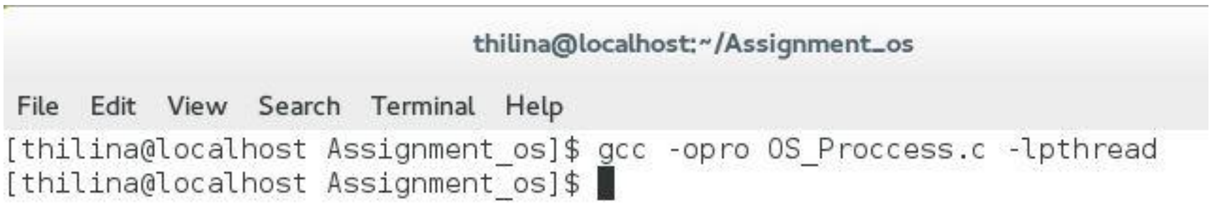
3.2 Threads

To compile **gcc –othr OS_thread.c –lpthread**

Then **./thr data1.txt 9 9**

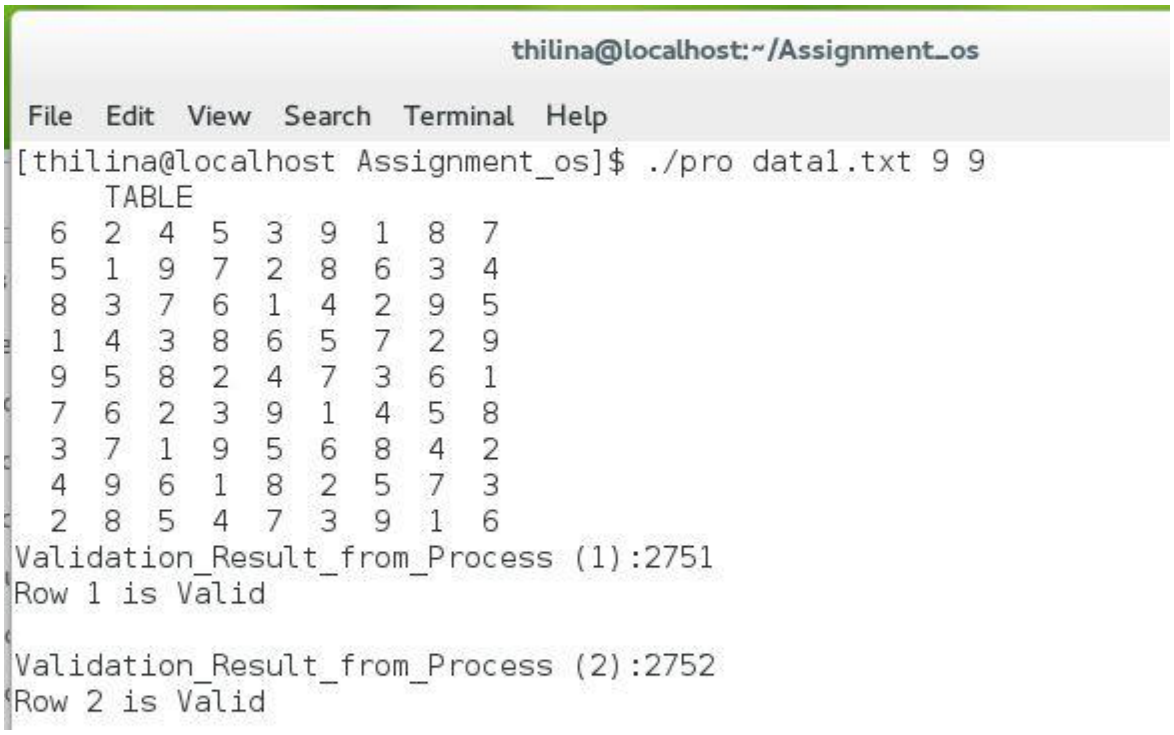
4. **Screenshots**

4.1 Processes


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

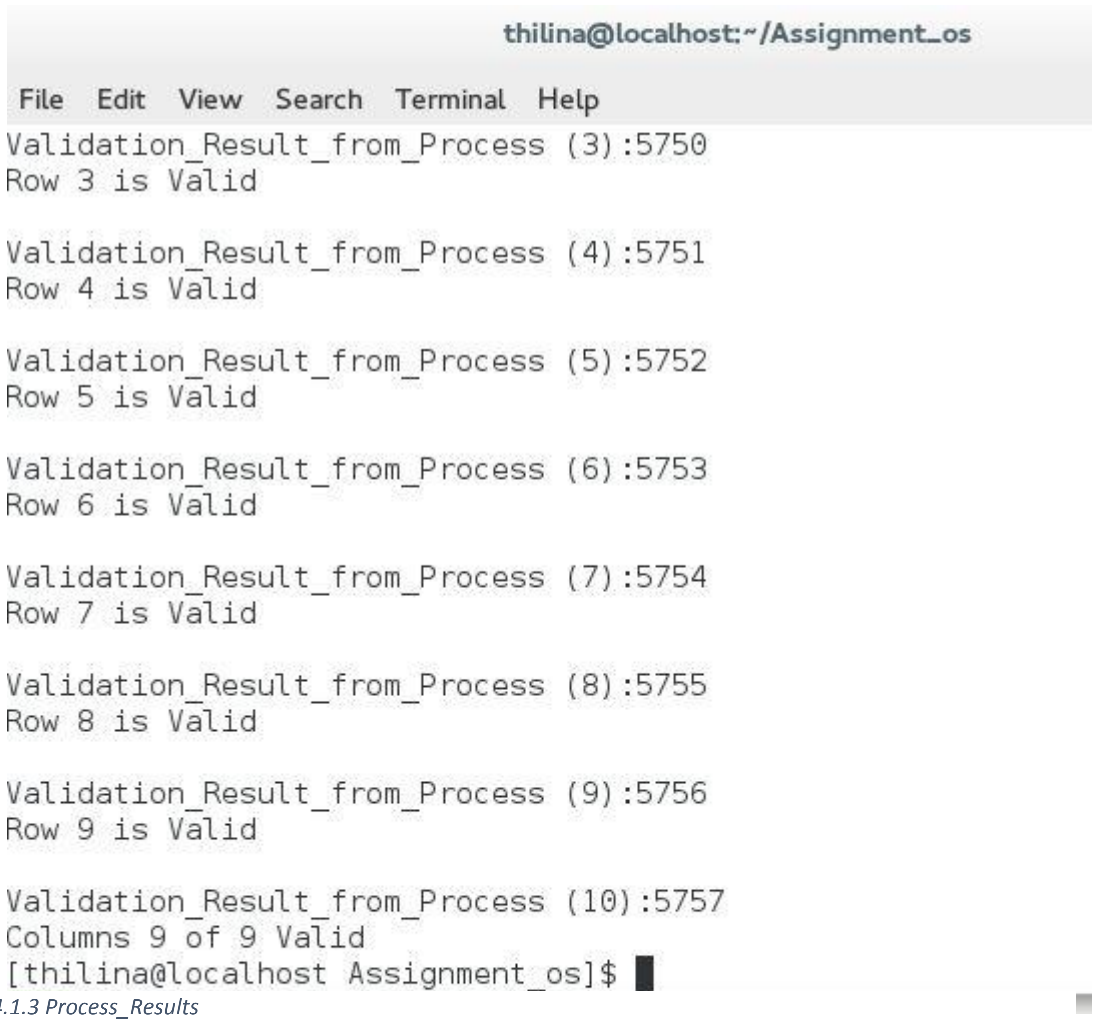
```
[thilina@localhost Assignment_os]$ gcc -opro OS_Proccess.c -lpthread
[thilina@localhost Assignment_os]$ 
```

4.1.1 Process Compile


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

```
[thilina@localhost Assignment_os]$ ./pro data1.txt 9 9
     TABLE
  6  2  4  5  3  9  1  8  7
  5  1  9  7  2  8  6  3  4
  8  3  7  6  1  4  2  9  5
  1  4  3  8  6  5  7  2  9
  9  5  8  2  4  7  3  6  1
  7  6  2  3  9  1  4  5  8
  3  7  1  9  5  6  8  4  2
  4  9  6  1  8  2  5  7  3
  2  8  5  4  7  3  9  1  6
Validation_Result_from_Process (1):2751
Row 1 is Valid

Validation_Result_from_Process (2):2752
Row 2 is Valid
```

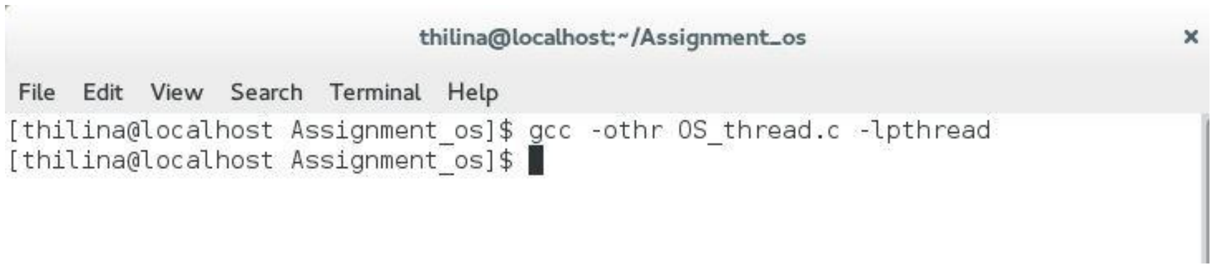
4.1.2 Process Execute + pass command args .


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

```
Validation_Result_from_Process (3):5750
Row 3 is Valid

Validation_Result_from_Process (4):5751
Row 4 is Valid

Validation_Result_from_Process (5):5752
Row 5 is Valid

Validation_Result_from_Process (6):5753
Row 6 is Valid

Validation_Result_from_Process (7):5754
Row 7 is Valid

Validation_Result_from_Process (8):5755
Row 8 is Valid

Validation_Result_from_Process (9):5756
Row 9 is Valid

Validation_Result_from_Process (10):5757
Columns 9 of 9 Valid
[thilina@localhost Assignment_os]$
```

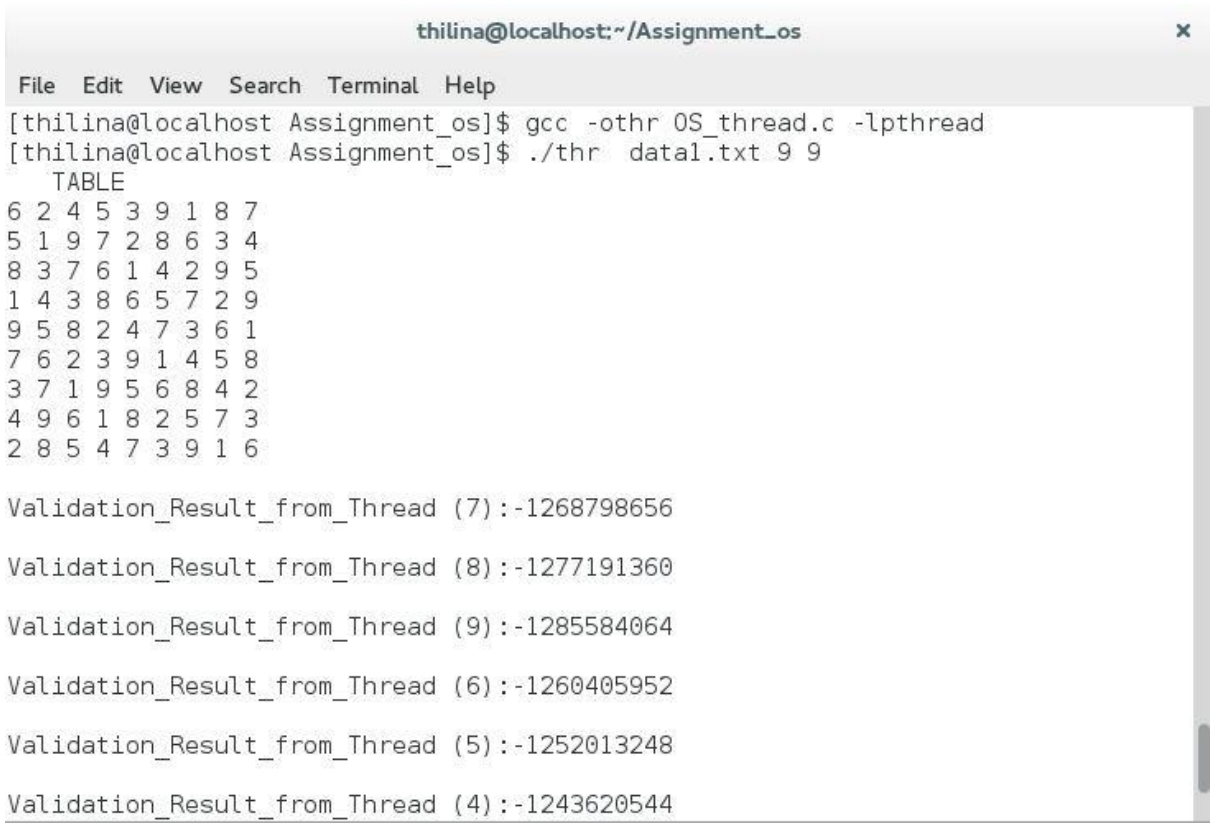
4.1.3 Process_Results

4.2 Threads


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

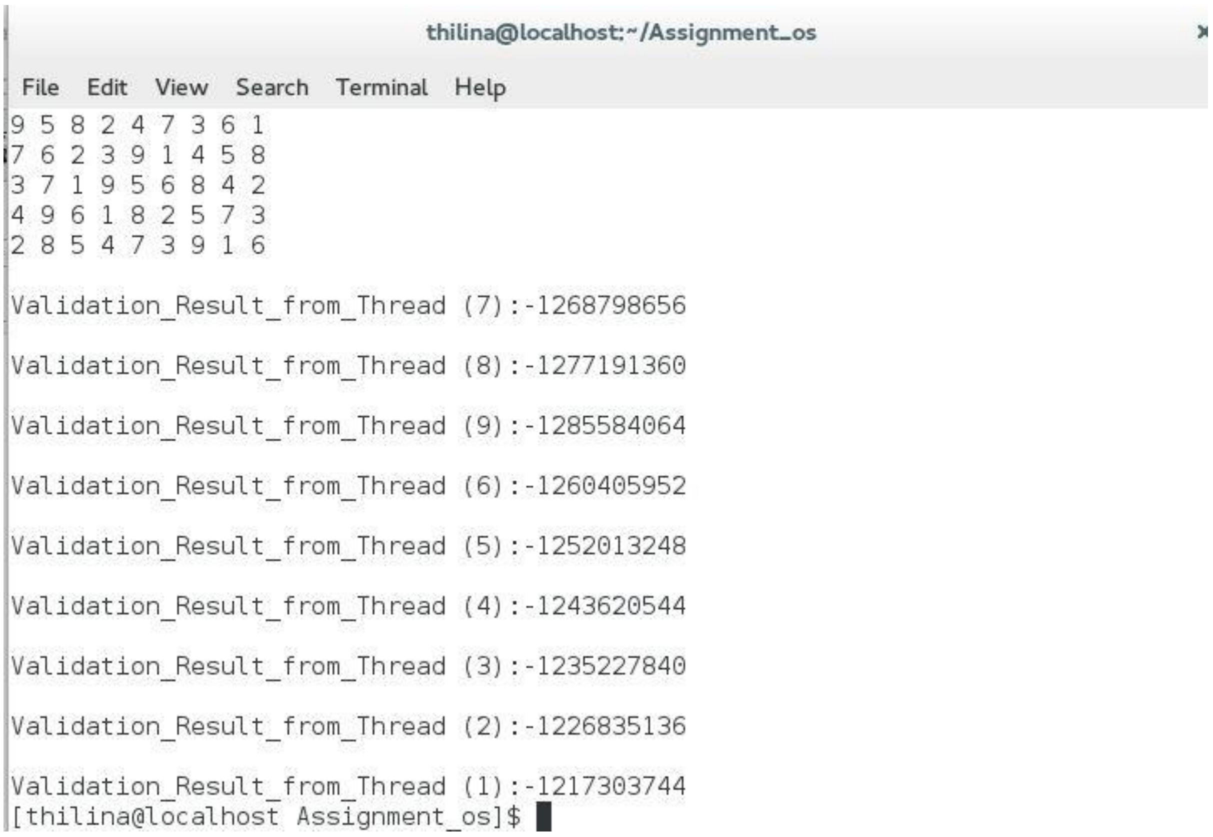
```
[thilina@localhost Assignment_os]$ gcc -othr OS_thread.c -lpthread
[thilina@localhost Assignment_os]$ 
```

4.2.1 Threads Compile


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

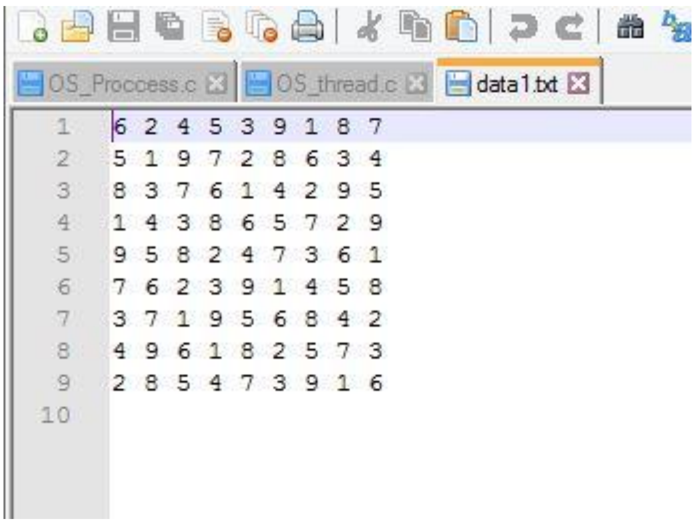
```
[thilina@localhost Assignment_os]$ gcc -othr OS_thread.c -lpthread
[thilina@localhost Assignment_os]$ ./thr  data1.txt 9 9
   TABLE
6 2 4 5 3 9 1 8 7
5 1 9 7 2 8 6 3 4
8 3 7 6 1 4 2 9 5
1 4 3 8 6 5 7 2 9
9 5 8 2 4 7 3 6 1
7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6

Validation_Result_from_Thread (7):-1268798656

Validation_Result_from_Thread (8):-1277191360

Validation_Result_from_Thread (9):-1285584064

Validation_Result_from_Thread (6):-1260405952

Validation_Result_from_Thread (5):-1252013248

Validation_Result_from_Thread (4):-1243620544
```

4.2.2 Excute+cmd arg


thilina@localhost:~/Assignment_os
File Edit View Search Terminal Help

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

Validation_Result_from_Thread (7):-1268798656

Validation_Result_from_Thread (8):-1277191360

Validation_Result_from_Thread (9):-1285584064

Validation_Result_from_Thread (6):-1260405952

Validation_Result_from_Thread (5):-1252013248

Validation_Result_from_Thread (4):-1243620544

Validation_Result_from_Thread (3):-1235227840

Validation_Result_from_Thread (2):-1226835136

Validation_Result_from_Thread (1):-1217303744
[thilina@localhost Assignment_os]$ 
```

4.2.3 Results



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

4.2.4 DATA INPUT.

5. **References**

1. http://www.unix.com/programming/172517-put-2d-array-shared-memory.html
2. Labsheets
3. http://stackoverflow.com/questions/29336955/sudoku-validator-in-c-using-threads-causing-segmentation-fault
4. http://stackoverflow.com/questions/41440365/2d-arrays-with-shared-memory
5. http://stackoverflow.com/questions/31188015/c-program-to-check-valid-sudoku