Part 1.1

To calculate time: Clock_gettime function is used as follows:

```
if( clock_gettime( CLOCK_REALTIME, &start) == -1 ) {
  perror( "clock gettime" );
  return ;
}
```

```
if( clock_gettime( CLOCK_REALTIME, &stop) == -1 ) {
  perror( "clock gettime" );
  return ;
}
```

```
accum = ( stop.tv_sec - start.tv_sec )
        + (double)( stop.tv_nsec - start.tv_nsec )
          / (double)BILLION;
```

Error handling is done as follows:

```
perror( "clock gettime" );
```

To execute code func() function is made where it calls Thr_B(),Thr_A(),Thr_C() functions.

These functions create thread as asked in the question.

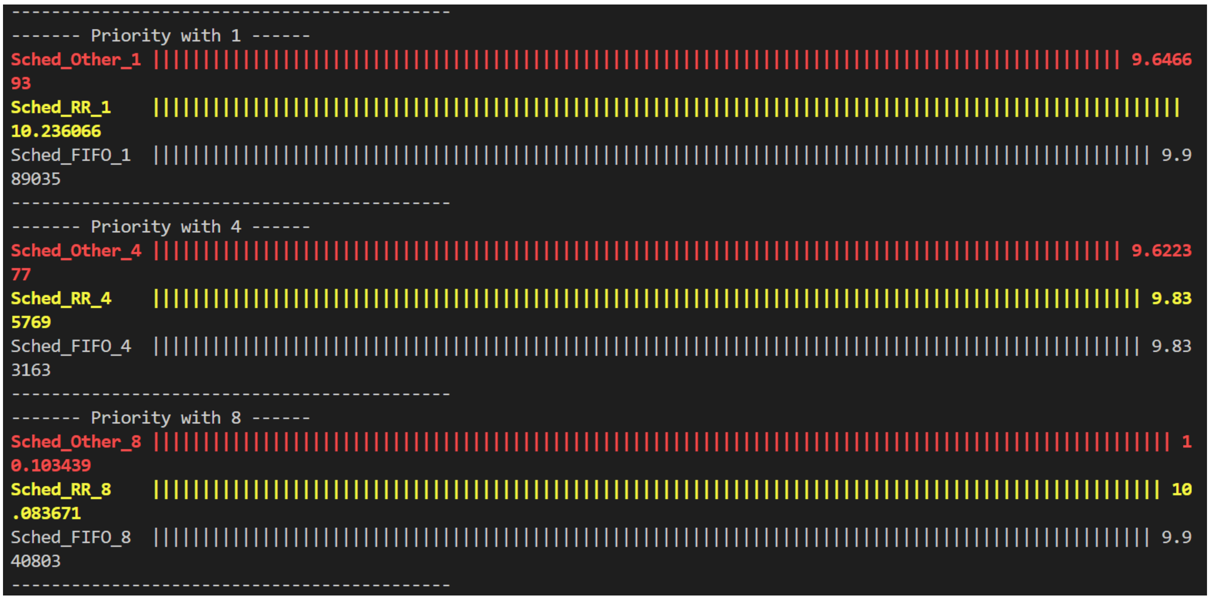
To print Histogram a function is made:

```
printHistogram(hist, 0.1);
```

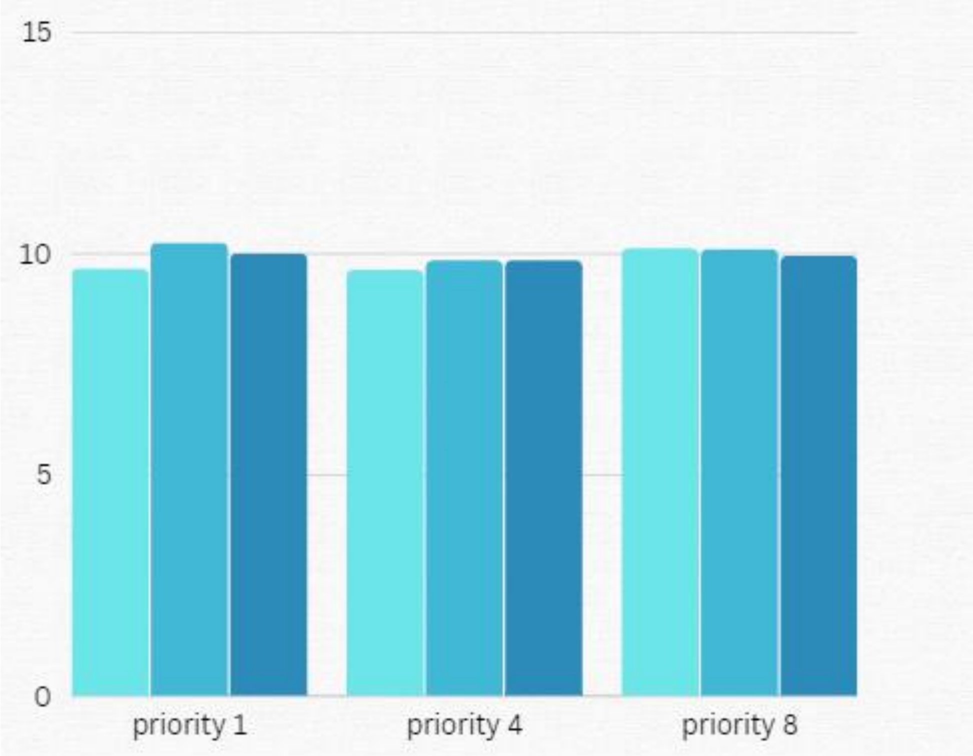
Array contains priority values as:

```
int Priority[3][3]={{1,1,1},{4,4,4},{8,8,8}};
```

Count A ,Count B, Count C functions are made to calculate time for a loop running from 1 to 2 to the power 32.



```
--------------------------------------------
------- Priority with 1 ------
Sched_Other_1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.6466
93
Sched_RR_1    ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
10.236066
Sched_FIFO_1  ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.9
89035
--------------------------------------------
------- Priority with 4 ------
Sched_Other_4 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.6223
77
Sched_RR_4    |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.83
5769
Sched_FIFO_4  |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.83
3163
--------------------------------------------
------- Priority with 8 ------
Sched_Other_8 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 1
0.103439
Sched_RR_8    |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 10
.083671
Sched_FIFO_8  ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 9.9
40803
--------------------------------------------
```



1st bar- SCHED_OTHER

2nd bar- SCHED_RR

3rd bar- SCHED_FIFO

Part 1.2

For this three bash script has been made as follows:

- Fork_Exec.c
- Makefile
- compilationofKernel1.sh
- compilationofKernel2.sh
- compilationofKernel3.sh

Main function makes 3 processes using fork() named as pid1,pid2,pid3.

To calculate time clock_gettime is used as:

```
if( clock_gettime( CLOCK_REALTIME, &start) == -1 ) {
perror( "clock gettime" );

}
execvp(args1[0],args1);

if( clock_gettime( CLOCK_REALTIME, &stop) == -1 ) {
perror( "clock gettime" );
```

To set priority sched_priority function is used as:

```
param2.sched_priority = 3;
```

To execute child process execvp function is used (family of exec):

```
execvp(args2[0],args2);
```

Error handling is done as follows:

```
perror( "clock gettime" );
```

```
printf("error creating process");
```

As asked in question waitpid is used to execute the child process till it ends completely.

```
int pid1_res= waitpid(pid1,NULL,0);
int pid2_res= waitpid(pid2,NULL,0);
int pid3_res= waitpid(pid3,NULL,0);
```

Part 2

Steps following @https://cool-dev.in/posts/How-to-Implement-a-System-Call-In-Linux/

Make kernel_syscall directory in source and make

kernel_syscall.c

artix Clone [Running] - Oracle VM VirtualBox

```
  GNU nano 7.0                                          kernel_syscall.c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(kernel_2d_memcpy,float**,matrix_1st,float**,matrix_2nd,int,row,int,column)
{

float matrix[row][column];

long copy_matrix=__copy_from_user(matrix,matrix_1st,row*column);

if (copy_matrix<0 || copy_matrix==row*column)

        return -EFAULT;
long returnvalue=__copy_to_user(matrix_2nd,matrix,row*column);

if (returnvalue<0 || returnvalue== row*column)
        return -EFAULT;

printk(KERN_INFO "Successfull : kernel_2d_memcpy executed");

return copy_matrix;

}
```

Makefile

Obj-y := kernel_syscall.c

-then compile kernel using make -j$(nproc)

And so on.

Test code I submitted in GC named->

copy2dmatrix.c

makefile