CLASS ASSIGNMENT 3
ID : 12341550

TASK 1

FILE : proc.h

in struct proc added shed_count and run_ticks

```
int shed_count;
int run_ticks;
```

TASK 2

FILE : proc.c

in allocproc() function before return p statement

```
p→shed_count = 0;
```

TASK 3

FILE : proc.c

in scheduler() function before context switch happens

```
p→shed_count++;
```

TASK 4

FILE : trap.c

in trap() function , in switch case T_IRQ0 + IRQ_TIMER:

```
if(myproc() && myproc()->state == RUNNING){
    myproc()->run_ticks++;
}
```

TASK 5

FILE : syscall.h

#define SYS_getstats 25

FILE : syscall.c

extern int sys_getstats(void);

[SYS_getstats] sys_getstats,

TASK 6

FILE : sysproc.c

```
int
sys_getstats(void)
{
  int *user_stats_ptr;

  if(argptr(0, (void*)&user_stats_ptr , 2 * sizeof(int)) < 0)
    return -1;
  struct proc * p = myproc();
  int kernel_stats[2];
  kernel_stats[0] = p->shed_count;
  kernel_stats[1] = p->run_ticks;

  if(copyout(p->pgdir , (uint)user_stats_ptr , (char*)kernel_stats ,
  sizeof(kernel_stats)) < 0)
    return -1;

  return 0;
}
```

TASK 7

FILE : user.h

```
struct procstats
{
    int count;
    int ticks;
};
...
int getstats(int * stats_array);
```

FILE : usys.S

```
SYSCALL(getstats)
```

TASK 8

FILE : statstest.c

```
#include "types.h"
#include "stat.h"
#include "user.h"
int main(void)
{
    int stats[2];
    int i;
    for (i = 0; i < 2; i++)
    {
        // When you pass ’stats ’, you are correctly passing a pointer to the
beginning of the array .
        if (getstats(stats) == 0)
        {
            // Access the elements using array indices .
            printf(1, "Scheduled %d times , ran for %d ticks \n", stats[0],
stats[1]);
        }
```

```
    else
    {
        printf(2, "getstats failed \n ");
    }
    sleep(10);
  }
  exit();
}
```

TASK 9

FILE : Makefile

_statstest

OUTPUT :