Assignment 2A (G26)

Task 1:

Added declaration of wait2 and set_prio in defs.h.
Initial processes are given priority '2' in exec.c.
In Makefile, SCHEDFLAG is passed to the compiler as given by the user. If not given, SCHEDFLAG is set to DEFAULT.
Set quanta to 5 so that preemption will be done every 5 clock ticks in param.h.
In proc.h file, added ctime, stime, retime, rutime, priority and tickcounter fields in proc struct and declared updatestatistics function which updates them for each process.
In proc.c file, implementation of all scheduling algorithms and functions like updatestatistics, wait2, set_prio functions is present.
In traps.c interrupts are handled through yield function according to different scheduling algorithms.

Task 2:

In this task, we implemented yield system call which returns 0 on every successful call and 1 otherwise.
For this, we did changes in syscall.c, syscall.h, sysproc.c, user.h, usys.S to successfully add our new system call.
The implementation was done in sysproc.c file.

Task 3.1:

For this task, we implemented a sanity program. This program takes an argument from the user and based on that number say n, 3*n processes are forked and for every child process we printed readytime, runtime and sleeping time of each one of them with the type of process i.e., CPU bound, S-CPU bound or I/O bound and also kept their aggregate sum in tot_time array.
For this wait2() function which we had implemented in assignment 2 was used. wait2() function provided all those required time statistics for every child process.
Output shown by the console:

```
init: starting sh
$ sanity 5
Cpu bound  , pid: 4, ready: 0, running: 0, sleeping: 0, turnaround: 0
S-Cpu bound, pid: 5, ready: 0, running: 0, sleeping: 0, turnaround: 0
Cpu bound  , pid: 7, ready: 0, running: 0, sleeping: 0, turnaround: 0
S-Cpu bound, pid: 8, ready: 0, running: 0, sleeping: 0, turnaround: 0
Cpu bound  , pid: 10, ready: 0, running: 0, sleeping: 0, turnaround: 0
S-Cpu bound, pid: 11, ready: 0, running: 0, sleeping: 0, turnaround: 0
Cpu bound  , pid: 13, ready: 0, running: 0, sleeping: 0, turnaround: 0
S-Cpu bound, pid: 14, ready: 0, running: 0, sleeping: 0, turnaround: 0
Cpu bound  , pid: 16, ready: 0, running: 0, sleeping: 0, turnaround: 0
S-Cpu bound, pid: 17, ready: 0, running: 1, sleeping: 0, turnaround: 1
I/O bound  , pid: 6, ready: 0, running: 0, sleeping: 100, turnaround: 100
I/O bound  , pid: 9, ready: 0, running: 0, sleeping: 100, turnaround: 100
I/O bound  , pid: 12, ready: 0, running: 0, sleeping: 100, turnaround: 100
I/O bound  , pid: 15, ready: 0, running: 0, sleeping: 100, turnaround: 100
I/O bound  , pid: 18, ready: 0, running: 0, sleeping: 100, turnaround: 100
Cpu bound stats  :
 Average ready time: 0
 Average running time: 0
 Average sleeping time: 0
 Average turnaround time: 0
S-Cpu bound stats:
 Average ready time: 0
 Average running time: 0
 Average sleeping time: 0
 Average turnaround time: 0
I/O bound stats  :
 Average ready time: 0
 Average running time: 0
 Average sleeping time: 100
 Average turnaround time: 100
$
```

Task3.2:

For this task we added a new program SMLsanity.c followed by adding it to UPROGS=/ list of Makefile.
This program too takes an integer argument from the user and then performs the required tasks.
For implementation of this we forked 21 processes and based on their pid we assigned priority to them.
Again with the help of wait2() system call we obtained required time statistics i.e., retime, runtime and stime and pid and with getctime() system call we got the creationtime and thus the termination time of the processes were calculated by adding ctime , retime, runtime and stime. After it, we printed the pid , priority and the termination time of the processes.
Parent process kept on spawning new child processes.

Output shown by console is:

```
init: starting sh
$ SMLsanity
Priority 1, pid: 4, termination time: 2657
Priority 2, pid: 5, termination time: 2656
Priority 3, pid: 6, termination time: 2656
Priority 1, pid: 7, termination time: 2656
Priority 2, pid: 8, termination time: 2657
Priority 3, pid: 9, termination time: 2656
Priority 1, pid: 10, termination time: 2656
Priority 2, pid: 11, termination time: 2656
Priority 3, pid: 12, termination time: 2656
Priority 1, pid: 13, termination time: 2656
Priority 2, pid: 14, termination time: 2656
Priority 3, pid: 15, termination time: 2656
Priority 1, pid: 16, termination time: 2656
Priority 2, pid: 17, termination time: 2656
Priority 3, pid: 18, termination time: 2656
Priority 1, pid: 19, termination time: 2656
Priority 2, pid: 20, termination time: 2656
Priority 3, pid: 21, termination time: 2656
Priority 1, pid: 22, termination time: 2656
Priority 2, pid: 23, termination time: 2656
Priority 3, pid: 24, termination time: 2656
$
```