REPORT

SCHEDULER - Priority Based

Implemented the priority based scheduler in xv6. The scheduler schedules the process with higher priority and in case there are two processes with current highest priority, round-robin is applied.

The scheduler always starts at the current process and searches the proctable for next process with highest priority starting from the current process thus ensuring that RR policy is followed for equal priority cases. The scheduler checks NPROC [number of processes] starting from the current process to determine the next process with the highest priority.

The set_priority system call finds the process with the given process id in the proctable and changes its priority. It uses the concept of yield syscall to reschedule if a process's priority is altered.

TEST

For testing the scheduler there is a program foo.c which forks a given number of processes according to the argument which is passed.
Each of these processes is assigned the default priority which is 60.
Now, with the ps command it prints the current process state, pid, and the priority. As all processes have priority 60, constantly running the ps command shows that the RUNNING process is constantly changing thus showing that RR is being followed in equal highest priority processes.

Also we can change the priority of a process using the set_priority system call from the sp command(called in sp.c program). If priority of a given process is reduced than we can check using the ps command that it is constantly runnning thus showing that highest priority process is always running.

COMPARISON AND CONCLUSION

- It is clearly evident that higher priority processes get more CPU time in this scheduling mechanism.
- The process with higher priority is preferred thus it guarantees more CPU time to higher priority process as compared to the original RR approach.