Xv6 Scheduling -Assignment 5

Abhishek Reddy Gangapuram-2018101028

1. run make qemu-nox <SCHEDULER>
2. run make clean after terminating

Waitx - Implementation

1. ctime is set in allocproc()
2. etime is set in exit()
3. rtime is incremented every time in ticks is incremented if the process is running
4. In waitx(int *wtime, int *rtime) we get wtime dynamically when requested. We use that wtime = (etime - ctime) – rtime
5. To test this use testwait
6.

SCHEDULING ALOGRITHMS

1. First Come First Serve (FCFS)
    1. Iterate through table and find the process which is RUNNABLE with minimum pid (!= 1 or 2) because if they are not ignored they will run forever
    2. disable yield()
    3. minimum pid is used as ctime could be same
2. Priority Based (PBS)
    1. this is same as fcfs but we use min priority instead of pid
    2. yield() is not disabled
    3. default priority is set to 60 for every process in allocproc()
    4. Iterate again through the table and run all RUNNABLE processes which have the same priority value in the struct proc as the value found in previous step
3. Multilevel Feedback Queue (MLFQ)
    1. Initially set all newly created processes to queue 0. Each process has curr_q variable in struct proc
    2. now use a variable to determine time quantums left to run in struct proc

3. Iterate through table and find if any runnable processes in queue 0, if none then check in queue 1, then in queue 2 and so on
4. If you find a process say in queue x. Then iterate through the whole table and run all the RUNNABLE processes which are in queue x
5. The previous step ensures that all the processes which are in the same queue are run in round robin fashion
6. Aging
    1. create a variable to store the time when is was last run
    2. now if it crosses a threshold reduce the priority of the queue
7. Time Quantums – Each time the tick is updated reduce the time quantums left for the RUNNING process and if it reduces to zero decrease the priority of the queue and update the time quantums

BENCHMARK

1. Run testprogram
2. this creates 10 processes with differnt times
3. each child process runs a order of 10^8 loop
4. and the parent exits
5. 1 time unit is 1 tick

Testing Results

TEST	**Default(Round Robin)**	**FCFS**	**MFQ**
Start time	862	291	336
End time	6733	6116	6120
Time elapsed	5817	5725	5784
ratio	1.016	1	1.01

Analysis

1. fcfs has least context switches and is fastest.
2. Mlfq is faster than default as it has less number of context switches than deafult