Modified xv6-riscv

1. Run the shell ,
   The shell supports Round Robin, First come First Serve, Priority Based Scheduling, Multi Level Feedback Queue Scheduling, Lottery Based Scheduling.
2. To use Round Robin Scheduling , Run the following command
   `make qemu`
3. To run FCFS or PBS or LBS or MLFQ, Run the following command,where `<choice>` is either FCFS or PBS
   `make qemu scheduler=<choice>`

Specification:

1.A)Syscall Trace:

1. Added `$U/_strace\` to `MAKEFILE`
2. Added new variable mask in `kernel/proc.h` and initialized it in function fork( ) in `kernel/sysproc.c`. To copy trace mask from parent to child process. `np->mask=p->mask`
3. Implemented a `sys_trace( )` function in `kernel/sysproc.c`

This function implements the new system call to assign the value for new variable mask

4. Modified `syscall( )` function in `kernel/syscall.c` to print trace output and `syscall_names` and `syscall_args` array to help with this
5. Created a user program in `user/strace.c` to generate the user-space stub for system calls.
6. Added a stub `user/usys.pl`, which causes the `makefile` to invoke the perl script `user/usys.pl` and produce `user/usys.s` and a system number to `kernel/syscall.h`

1.B)Alarm test

1. Implemented a `sigalarm() sigreturn()` function in `kernel/sysproc.c`
2. Added in `alarm.c`

2.Scheduling

1. First Come First Serve(FCFS):

FCFS selects the process with the lease creation time, which is the tick number corresponding to the time the process was created. The process is executed until it is terminated.

1. Modified the `MAKEFILE` to support SCHEDULER macro for the compilation of the specified scheduling algorithm. Here
2. Added `create_time` to `struct proc` in `kernel/proc.h`.

3. Initialized create_time to 0 in allocproc function in kernel/proc.c
4. Implemented scheduling functionality in scheduler( ) function in kernel/proc.c,
Where the process with least create_time is selected from available processes.
5. Disable yield from kernel/trap.c in order to prevent preemption of the process after the clock interrupts in FCFS .

2.Lottery Based Scheduler(LBS):

Lottery Based Scheduling is a preemptive scheduler that assigns a time slice to the process randomly in proportion to the number of tickets it owns. That is the probability that the process runs in a given time slice is proportional to the number of tickets owned by it.
Here
1. Added tickets to struct proc in kernel/proc.h.
2. Initialized tickets to 1 in allocproc function in kernel/proc.c
3. Implemented scheduling functionality in scheduler( ) function in kernel/proc.c,
Where the process with goldenticket(TICKECT) is selected from available processes.

3.Priority Based Scheduler(PBS):

PBS is a non-preemptive Priority Based scheduler that chooses the process with the highest priority of execution. When two or more processes have the same priority, the number of times the process has been scheduled is used to determine the priority. In case the tie still remains, the start-time of the processes are used to break the tie, with the processes having a lower start-time being assigned a higher priority.

1. Added variables staticPriority, rTime, create_time, nosch, and sleeptime to struct proc in kernel/proc.h.
2. Initialized the above variables with default values in allocproc( ) function in kernel/proc.c.
3. Added the scheduling functionality for PBS.
4. Added set_priority( ) function in kernel/proc.c.
5. Created a user program user/setpriority.c.
6. Added sys_set_priority( ) system call in kernel/sysproc.c.

4.Multi Level Feedback Queue(MLFQ):

Multi level feedback queue is a simplified preemptive MLFQ scheduler that allows processes to move between different priority queues based on their behavior and CPU bursts.

- If a process uses too much CPU time, it is pushed to a lower priority queue, leaving I/O bound and interactive processes in the higher priority queues.
- To prevent starvation, implement aging.

1. Added new variables struct proc to store the priority, allocated time, times dispatched, time added to queue, and time spent in each queue
2. Initialized the above variables with default values in `allocproc( )` function in `kernel/proc.c`.
3. Created 5 queues of different priority
4. Added the scheduling functionality for MLFQ.
5. Edited `scheduler( )` in kernel/proc.c to run the process with the highest priority.
6. Edited `kerneltrap( )` and `usertrap( )` in kernel/trap.c to yield when process has exhausted its time slice

Performance comparison:

Scheduler	Average waiting time	Running time
Round Robin	**162**	**15**
First Come First Serve	**136**	**30**
Lottery based scheduler	**130**	**15**
Priority based scheduler	**130**	**15**
Multi level feedback queue	**160**	**15**