Scheduling Algorithms : Report

FCFS:

In FCFS we compare the creation time of each process (which is stored is proc->ctime) and while scheduling choose the process with the lowest creation time, context switch on the CPU to run the process.
It is a non Preemptive algorithm and hence yield is ignored here, as process needs to wait because of convoy effect it results in larger average wait time for processes.

PBS:

In PBS we compare the process with the highest Dynamic priority (lowest value). Static priority (set default to 60) can be changed by the command line function setpriority [new_priority] [pid] which also sets niceness to be 5. Dynamic priority is calculated by the function

DP = max(0, min(100, SP - niceness + 5))

where niceness is calculated as the percentage of time process spends in sleeping since last it was called, (in proc struct last_run and last_sleep were created and stored to calculate the percentage run).

For setpriority user function int set_priority(int new_priority, int proc_id) syscall is used.

When the priority of the process is increased it is preempted and rescheduling of highest priority process is done.

MLFQ:

5 Queues are created, implementing circular queue ADT, pushq, popq functions are also implemented to push and pop the processes in the respective priority_queue.

Process is given time quantum 2^i if it is present in ith queue and preempted after it’s time quantum expires and added into a lower priority_queue.

Aging function is implemented in update_time function which calculated the run and wait time for the process, which is called in clockintr() in kernel/trap.c and to update value at each tick.

Also aging is done to prevent starvation, if waiting time of the process in queue is grated than 64 than it is promoted to higher priority queue and it is checked after every tick.

Also the process is preempted if a process enters a lower priority_queue and new process is scheduled, also if process is sleeping, waiting for I/O it is removed from the queue.

Preemption is implemented in yield() function.

LBS:

The process is allotted the cpu randomly in proprotion to the number of tickets it owns.We are summing the total tickets and then generating a random number to involve randomness and then allocating the cpu to that process tickets interval in which the random no lies . So , the processes are scheduled in proportion to the no of tickets it owns . Also we need a variable tickets for a process (declared in proc.h) and set to 1 in allocproc , the child is also given the same no of tickets as parent in fork . We are using settickets system call to change the no of tickets of a process .

Also the scheduling is pre-emptive so a process will yield if a process with higher no of tickets comes into the queue.