CS330 - Assignment

Nishi Mehta 200645
Mohd Umam 200595
Maurya Jadav 200567
Shubham Kumar 200966

October 26, 2022

1. Implementation of **SCHED_NPREEMPT_SJF**

   Once we enter the condition for SJF scheduling policy, First we iterate through all the RUNNABLE processes and check if the RUNNABLE process is not created by forkp(). Then that process is scheduled first and we break out of the loop else we find the process with minimum *estimate* among all the RUNNABLE processes and the state of that process is changed from RUNNABLE to RUNNING by the scheduler(). When a particular process starts running, we compute the *xticks* and store it in *CPU_burst_start* which is a parameter in struct proc. *CPU_burst_end* is computed by getting the *xticks* at the time when the process transits from RUNNING to either RUNNABLE, SLEEPING or ZOMBIE state. *CPU_burst* is the difference between *CPU_burst_start* and *CPU_burst_end* and this *CPU_burst* is used in each iteration to update the *estimate* of the process.

   The SJF algorithm is implemented in scheduler() system call, while the *CPU_burst* and *estimate* are computed in yeild(), sleep() and exit() system calls. All four are present in *proc.c*.

2. Implementation of **SCHED_PREEMPT_UNIX**

   In the UNIX scheduler, we maintain two for loops. First *for* loop updates *CPU_usage* and *priority*(Dynamic Priority) of each RUNNABLE process. The second *for* loop checks if any process is a non-batch process if found, then directly transit the process from RUNNABLE to RUNNING state else, the process with minimum *priority* is converted from RUNNABLE to RUNNING state. CPU_usage of each process is incremented in yeild() and sleep() system calls by *SCHED_PARAM_CPU_USAGE* and *SCHED_PARAM_CPU_USAGE*/2 respectively.

   All the modifications are made in same functions same as of SJF scheduling policy.

	File	Algorithm	Batch execution time	Average turn-around time	Average waiting time	Completion time			CPU bursts				CPU burst estimates				CPU burst estimates error	
						avg	max	min	count	avg	max	min	count	avg	max	min	count	avg
1	batch1.txt	SCHED_NPREEMPT_FCFS	16028	7371	202	9172	16029	2003	-	-	-	-	-	-	-	-	-	-
		SCHED_PREEMPT_RR	15809	13032	11737	14802	15810	14792	-	-	-	-	-	-	-	-	-	-
	batch2.txt	SCHED_NPREEMPT_FCFS	14043	6500	132	7672	14044	1304	-	-	-	-	-	-	-	-	-	-
		SCHED_PREEMPT_RR	15522	12879	11601	15514	15523	15501	-	-	-	-	-	-	-	-	-	-
	batch7.txt	SCHED_NPREEMPT_FCFS	15866	6981	259	9298	15867	2577	-	-	-	-	-	-	-	-	-	-
		SCHED_PREEMPT_RR	15184	12892	11608	15171	15185	15159	-	-	-	-	-	-	-	-	-	-
3	batch4.txt	SCHED_NPREEMPT_FCFS	11337	51154	130	6310	11383	1286	-	-	-	-	-	-	-	-	-	-
		SCHED_NPREEMPT_SJF	11843	9545	101	10442	11844	999	90	4643	11844	1	90	36413	10929	1	90	10100
4	batch5.txt	SCHED_PREEMPT_RR	15767	14843	13357	15756	15768	15743	-	-	-	-	-	-	-	-	-	-
		SCHED_PREEMPT_UNIX	18511	9896	408	13184	18512	4090	-	-	-	-	-	-	-	-	-	-
	batch6.txt	SCHED_PREEMPT_RR	12033	10963	9677	12025	12034	12019	-	-	-	-	-	-	-	-	-	-
		SCHED_PREEMPT_UNIX	15786	9540	145	10994	15787	1456	-	-	-	-	-	-	-	-	-	-

3. Observation
   Turn-around time of round robin is greater than FCFS because in FCFS processes complete near about sequentially where as in round robin it complete parallel. Waiting time of FCFS is very less.

4. Question 3 observation:-
   batch2.txt :- ratio = 3.39
   batch3.txt :- ratio = 1.2