CS330A: Design Assignment 2

Group 15

Aarchie [200004] Kembasaram Nitin [200505]
Harsh Jain [200412] Udit Prasad [201055]

Implementation

While going through inner loop of the scheduler, we are checking if the scheduling algorithm is changed to either SJF or Unix scheduler or not. If it is changed we break from the loop and go into the implementation of whichever scheduling policy is invoked.
For Non preemptive FCFS and SJF, we have disabled timer interrupt by checking for these scheduling policies in *usertrap* and *kerneltrap* functions and not calling yield in that case.

SCHED_NPREEMPT_SJF

Run a loop and go through all the process :

1. If there is any runnable process which is not a batch process, schedule it right away.
2. For batch processes, keep track of minimum next cpu burst and the associated process for all runnable processes and when we are done with all the process, schedule the process with least next cpu burst.

Updation of next cpu burst of a process :

- Keep a variable *cpu_burst_start*, which we change to the value of *ticks* whenever any batch process is scheduled (i.e. just before swtch call) so to keep track of current cpu burst's start time.
- After that whenever a process goes into *yield()* or *sleep()*, calculate t_n(current cpu burst) using $t_n = currentticks - cpu\_burst\_start$
- Then use s_n ($p \rightarrow next\_cpu\_burst$) as $p \rightarrow next\_cpu\_burst$ is not updated yet and t_n to calculate s_{n+1} using the formula and update $p \rightarrow next\_cpu\_burst$ with it.

SCHED_PREEMPT_UNIX

- Run a loop and go through all the non batch processes and if there is any runnable one, schedule it right away.
- Then, Run a loop over all the process and if it is a batch process and runnable, then :
  - update $p \rightarrow cpu\_usage = p \rightarrow cpu\_usage/2$
  - update $p \rightarrow priority = p \rightarrow base\_priority + cpu\_usage/2$
  - Look for process with minimum priority value and update it accordingly.
- Updation of cpu usage is done in yield and sleep system calls as below:
  - sleep: cpu_usage += 100
  - yield: cpu_usage += 200

Evaluation

One statistics comparison is attached for each problem but result may differ everytime in small range.

1. Between FCFS and RR

   (a) batch1.txt

Statistics		
Stat	FCFS	RR
Batch Execution time	9469	9680
Avg. Turnaround time	9467	9660
Avg. Waiting time	8515	8687
Avg. Completion time	9931	10358
Max. Completion time	9934	10306
Min Completion time	9928	10378

   All the stats are more or less same in this case as the process has sleep call and this leads to take-away of cpu from process which reached first.
   But if we see the start time of all the processes, it is more or less same in Round Robin but has huge difference in FCFS. It clearly shows how fair is Round Robin and how unfair is FCFS.

   (b) batch2.txt

Statistics		
Stat	FCFS	RR
Batch Execution time	9730	9678
Avg. Turnaround time	9728	9664
Avg. Waiting time	8755	8696
Avg. Completion time	10247	10210
Max. Completion time	10260	10213
Min Completion time	10227	10208

   Here also, in Round Robin all the processes have very less difference in start time whereas huge difference in case of FCFS. Where as, average turnaround time as well as waiting time is approximately same for both Round Robin and FCFS.

   (c) batch7.txt

Statistics		
Stat	FCFS	RR
Batch Execution time	9639	9626
Avg. Turnaround time	5320	9607
Avg. Waiting time	4356	8644
Avg. Completion time	5869	10191
Max. Completion time	10190	10210
Min Completion time	1538	10157

   - Observation - In FCFS, firstly process with lower pid start and executes and then only next process starts as it is Non preemptive and testloop4.c has no sleep() or yield() call, which results in domination of cpu by process which reaches first
   - **Waiting time** - Avg. Waiting time is better in FCFS as some process do not wait for a long time as they complete soon but in Round Robin, all the process wait till last and this leads to better avg. waiting time in case of FCFS.
   - **Turnaround time** - Avg. Turnaround time is also better in case of FCFS because some process have very low turnaround time but its not the case in RR where every process have huge and same turnaround time.
     We also know turnaround time = waiting time + burst time + sleep time. This is also justified as both waiting time and turnaround time are better in FCFS.

- **Completion time**: Difference between max completion and min completion time is very high in case of FCFS whereas almost same in case of RR which shows how unfair is FCFS i.e. some process completes very fast and some wait till very long.

2. Between CPU burst estimation error of batch2.txt and batch3.txt

   - For batch2 Avg. CPU burst estimation error is close to 60 and in batch3, it drops down close to 20.
   - For batch2, Avg. cpu burst length = 177, Avg estimated CPU burst length = 143
     For batch3, Avg. cpu burst length = 192, Avg estimated CPU burst length = 182
     We can clearly see that batch3 has less difference in estimated and actual averages in comparison to batch2.

   Reason - Exponential averaging technique gives more accurate results for cpu burst time after sufficiently large number of cpu bursts as batch3 runs testlooplong and batch2 runs testloop2 and testlooplong has very high number of cpu bursts.

3. Between FCFS and SJF on batch4.txt

Statistics		
Stat	FCFS	SJF
Batch Execution time	7119	7257
Avg. Turnaround time	7116	5321
Avg. Waiting time	6405	4595
Avg. Completion time	7746	5985
Max. Completion time	7744	7923
Min Completion time	7750	3742

   - Observation- In SJF, All the processes scheduled for once and then all the odd pid process (testloop3) scheduled till they complete and after that the process with even pid (testloop2) scheduled.
     Reason- testloop3 has smaller cpu bursts compared to testloop2 and thats why SJF picks the testloop3 process before testloop2.
   - **Waiting time**: Avg waiting time is less for SJF as half of the processes completes early and dont wait much but in case of FCFS, all the process completes at almost same time because after each outer loop iteration, it gives up the cpu and process with next pid takes the cpu (and the loop continues) . So all processes completes at almost same time.
   - **Turnaround time**: Avg turnaround time is also less in case of SJF. It is because half of the processes have very less turnaround time which have small cpu bursts(testloop 3). But as explained above, in FCFS, all the process takes huge amount of time to complete.
     We also know turnaround time = waiting time + burst time + sleep time. This is also justified as both waiting time and turnaround time are better in SJF.
   - **Completion time**: There is large difference in max completion time and min completion time for SJF as the odd pid processes completes first and even ones later. But FCFS has very small difference in it as explained above.

4. Between Round Robin and Unix Scheduler on :

   (a) batch5.txt

Statistics		
Stat	RR	Unix
Batch Execution time	9584	9621
Avg. Turnaround time	9558	6315
Avg. Waiting time	8594	5347
Avg. Completion time	10068	7062
Max. Completion time	10089	10369
Min Completion time	10009	3591

- In Unix Scheduler, start time and end time of processes have very large time difference than that of RR.
  Reason - In Unix Scheduler, first the processes with higher pid i.e. low priority value ones are scheduled then after they get completed, some other processes with higher priority value starts getting scheduled and in this manner scheduling took place. Whereas in RR, all the processes got fair chance and all of them started in a difference of small time interval and also completed in same fashion.
- **Turnaround time**: Unix scheduler has better avg. turnaround time as higher priority ones get scheduled and completed early and hence did not wait till end that reduced their turnaround time and hence less avg. turnaround time as compared to RR where each process completed more or less together.
- **Waiting time**: Unix scheduler has better avg. waiting time as compared to RR. Reason being the higher priority process do not wait much as they complete early and hence less avg. waiting time.
- **Completion time**: Avg. Completion time is better in Unix scheduler due to same reasons explained above. Also there is huge difference in max completion time and min one which also justifies the fact that some process (high priority) completes soon and some later.

(b) batch6.txt

Statistics		
Stat	RR	Unix
Batch Execution time	9436	9324
Avg. Turnaround time	9421	6102
Avg. Waiting time	8478	5169
Avg. Completion time	10113	6616
Max. Completion time	10127	9839
Min Completion time	10093	3279

All the observations will be similar to 4.a but there will be some differences because in batch5, it is testloop1 whereas in batch6, it is testloop2. testloop1 has sleep calls whereas testloop2 has yield calls. This will lead to change in cpu usage of the processes (yield results in +200 and sleep results in +100) and thus a change in priority. So order will be different for batch5 and batch6 and also the stats .
However comparison with Round Robin will be similar as 4.b