CS342 Operating Systems

Project 2 Report

Ege Türker 21702993 Section 1

Instructor

İbrahim Körpeoğlu

Spring 2021

1) minB = 100 avgB = 200 minA = 100 avgA = 500

Average waiting time for 10 Threads with 10 Bursts each

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	20.698	23.528	1.838	10.270
2	12.219	11.481	5.025	9.969
3	16.221	5.716	8.226	10.379
4	10.250	15.389	11.754	8.749
5	18.116	11.434	14.698	17.796
6	12.749	17.384	18.155	13.949
7	12.883	17.578	21.619	19.284
8	16.084	2.719	24.651	15.639
9	14.921	25.527	28.749	17.406
10	14.528	7.930	31.233	19.875
Avg of all threads	14.867	13.869	16.595	14.332

2) minB = 100 avgB = 200 minA = 100 avgA=500

Average waiting time for 5 Threads with 20 Bursts each

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	13.889	18.690	2.737	12.028
2	17.642	17.865	8.440	16.766
3	16.806	11.181	13.886	15.217
4	14.810	22.150	18.664	18.033
5	10.757	4.986	22.580	20.269
Avg of all threads	14.781	14.974	13.261	16.463

3) minB = 100 avgB = 200 minA = 1000 avgA = 1500

Average waiting time for 10 Threads with 10 Bursts each

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	16.740	10.033	1.314	7.223
2	18.080	16.485	3.640	14.536
3	17.881	5.585	6.064	10.325
4	17.240	10.032	9.500	14.511
5	17.165	8.516	11.424	16.303
6	20.363	14.389	14.855	16.049
7	18.116	21.417	17.271	20.027
8	18.053	16.916	20.222	18.411
9	14.349	14.108	24.467	21.553
10	16.353	10.729	27.719	16.139
Avg of all threads	17.434	12.821	13.648	15.508

4) minB = 100 avgB = 200 minA = 1000 avgA = 1500

Average waiting time for 5 Threads with 20 Bursts each

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	15.889	17.032	3.304	8.023
2	15.526	11.083	9.330	15.532
3	12.496	6.533	14.419	15.662
4	11.398	16.097	20.521	15.826
5	11.311	19.739	27.147	23.063
Avg of all threads	13.324	14.097	14.944	15.613

5) minB = 200 avgB = 500 minA = 100 avgA = 500

Average waiting time for 10 Threads with 10 Bursts each

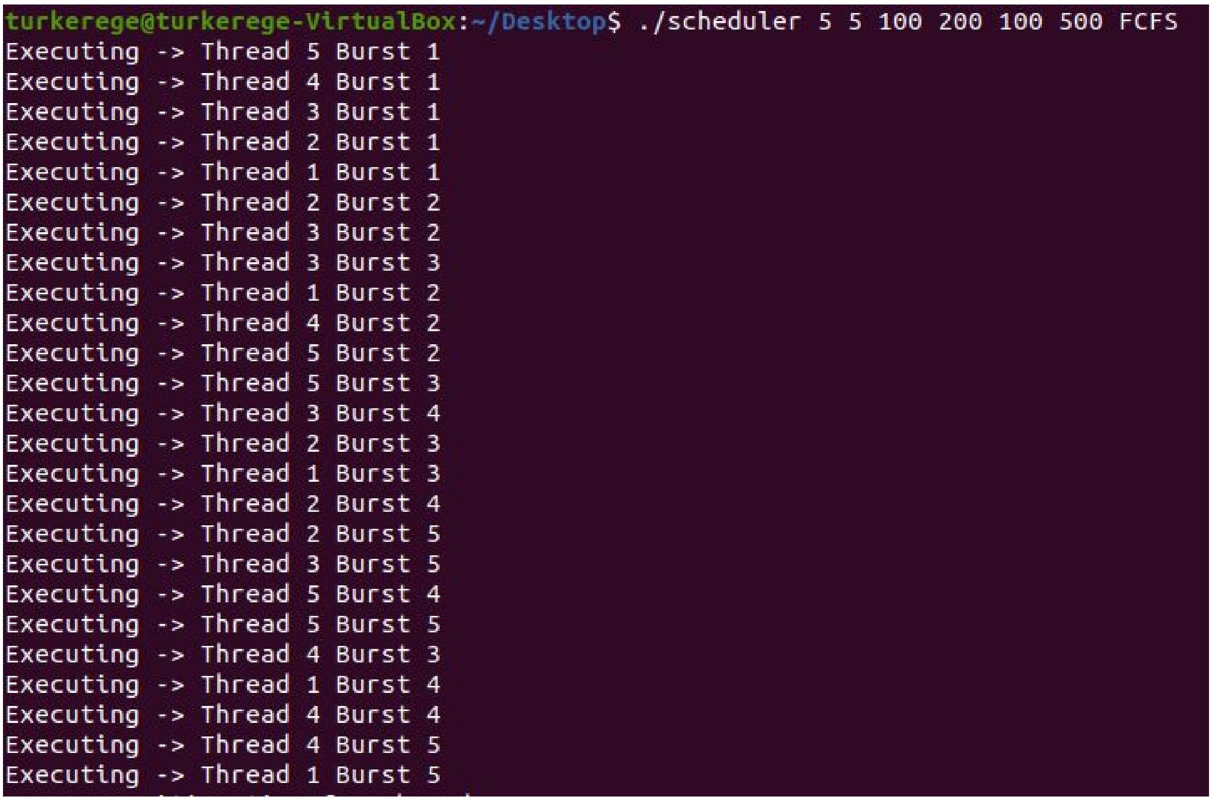
Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	45.737	13.399	3.176	26.988
2	29.709	60.790	11.347	28.272
3	34.721	31.980	18.247	24.948
4	40.836	28.402	22.474	26.056
5	31.729	28.565	29.511	32.482
6	47.189	39.414	36.617	37.712
7	26.143	17.989	44.090	42.074
8	30.439	43.196	50.512	54.423
9	27.104	57.608	57.847	47.993
10	31.651	59.112	64.871	53.887
Avg of all threads	34.526	38.045	33.869	37.483

6) minB = 200 avgB = 500 minA = 100 avgA = 500

Average waiting time for 5 Threads with 20 Bursts each

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	39.942	21.176	5.394	24.315
2	31.189	32.885	18.282	27.239
3	30.658	21.996	31.783	35.476
4	35.537	52.480	43.176	32.746
5	45.725	22.941	56.175	45.815
Avg of all threads	36.610	30.296	30.962	33.118

- Between experiments 1 and 2 the number of threads was changed but the total number of bursts was 100 for both of them. The resulting average of waiting times did not change significantly across all threads. Using 10 threads instead of 5 did not change the waiting times since the threads were used for production and the program was limited by the server thread.
- Across all experiments, threads with smaller index numbers always had smaller wait times when PRIO scheduling was used. This is expected since they are prioritized in that algorithm.
- Across all experiments, threads with smaller index numbers usually had smaller wait times when VRUNTIME scheduling was used. The effect was not as significant as PRIO since VRUNTIME also depends on the previous runtimes of the bursts alongside the thread index.
- There were outlier values in experiments. For example in experiment 1, thread 8 average waiting time for SJF was 2.719 seconds compared to the average of all threads 13.869 seconds. However in other experiments with 10 threads (experiment 3 and 5) such an outlier value for thread 8 was not observed in SJF. Those values were actually higher than average of all threads (16.916 > 12.821, 43.196 > 38.045). Thus it can be concluded that the outlier values that are not consistently reproduced are due to the random generation of burst times and not related to the algorithms used. Other examples to this are: experiment 2 - thread 5 - SJF, experiment 1 - thread 3 - SJF, experiment 5 - thread 7 - SJF.
- Outlier values were only observed for SJF and not for FCFS. The reason for this is that SJF relies on the burst length while picking the burst and thus it relies on random number generation. However, in FCFS, early bursts of a thread are always executed early and the late bursts are always executed late, creating average values. An example execution is presented at the bottom, showing the said execution pattern.

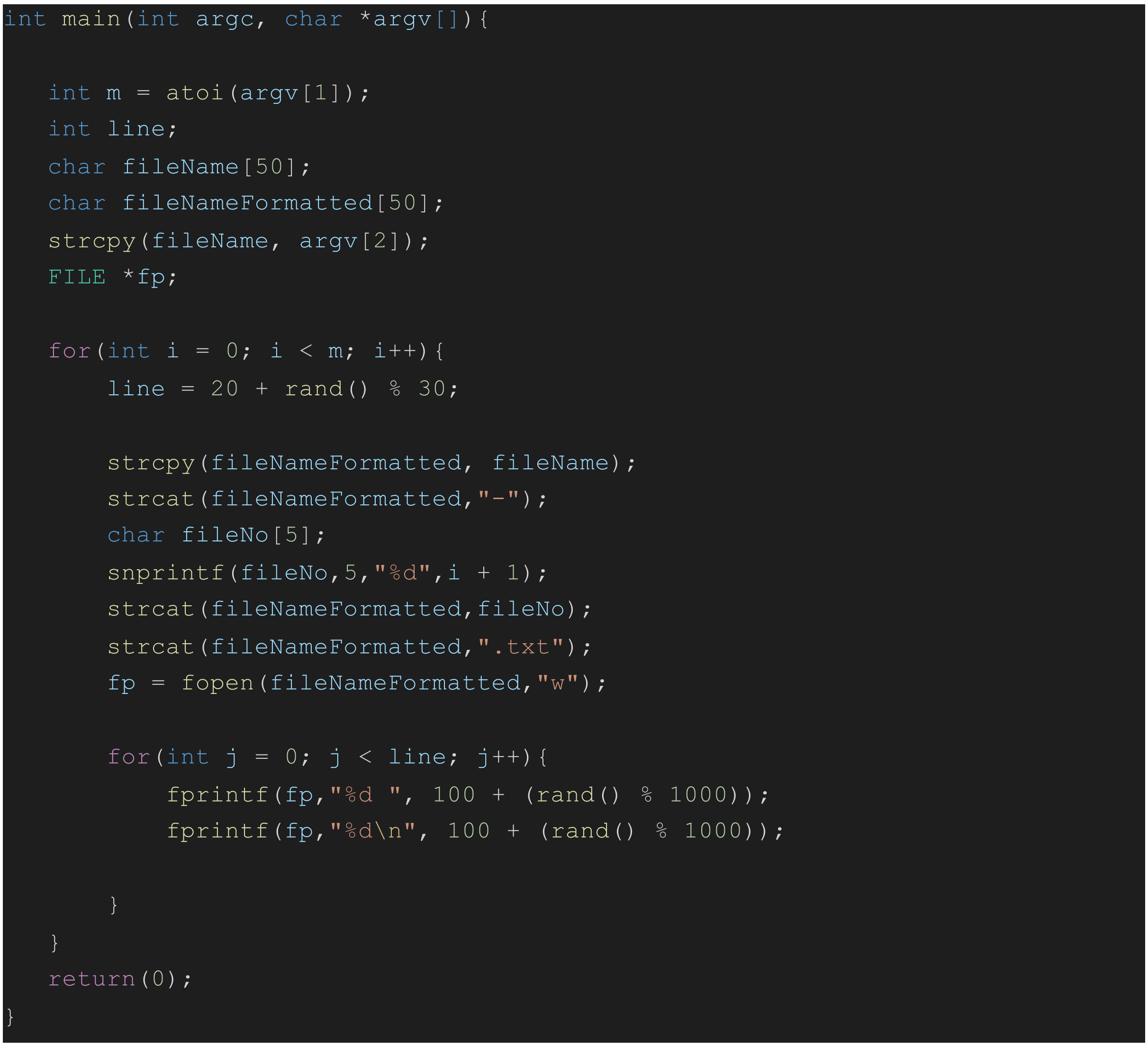


```
turkerege@turkerege-VirtualBox:~/Desktop$ ./scheduler 5 5 100 200 100 500 FCFS
Executing -> Thread 5 Burst 1
Executing -> Thread 4 Burst 1
Executing -> Thread 3 Burst 1
Executing -> Thread 2 Burst 1
Executing -> Thread 1 Burst 1
Executing -> Thread 2 Burst 2
Executing -> Thread 3 Burst 2
Executing -> Thread 3 Burst 3
Executing -> Thread 1 Burst 2
Executing -> Thread 4 Burst 2
Executing -> Thread 5 Burst 2
Executing -> Thread 5 Burst 3
Executing -> Thread 3 Burst 4
Executing -> Thread 2 Burst 3
Executing -> Thread 1 Burst 3
Executing -> Thread 2 Burst 4
Executing -> Thread 2 Burst 5
Executing -> Thread 3 Burst 5
Executing -> Thread 5 Burst 4
Executing -> Thread 5 Burst 5
Executing -> Thread 4 Burst 3
Executing -> Thread 1 Burst 4
Executing -> Thread 4 Burst 4
Executing -> Thread 4 Burst 5
Executing -> Thread 1 Burst 5
```

- Between experiments 1-3 and 2-4 all values were kept the same except minA and avgA. However the change in burst generation frequency did not significantly change the average wait times. avgA was increased from 500 to 1500 and minA was increased from 100 to 1000. The change in average of all threads was not greater than 10-15%. This is most probably because of the single threaded server thread not reaching the speed of thread generation.

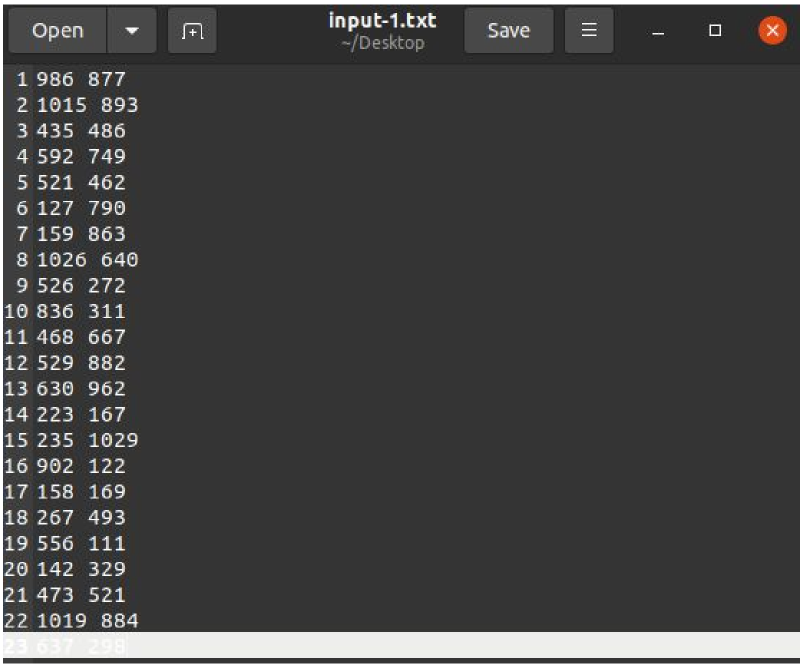
- Between experiments 1-5 and 2-6 all values were kept the same except minB and avgB. This resulted in a significant increase in average wait times. That was a natural and expected result, since minB and avgB values determine the lengths of the generated bursts. Increasing minB from 100 to 200 and avgB from 200 to 500, increased the average waiting times to 250-300% of the first experiments.

7) Another experiment was performed with the -f option of the program. With the help of the following code, 10 input files with randomized number of bursts, burst lengths and inter arrival times were generated. These random files had burst numbers between 20-30, burst length and inter arrival times between 100-1100ms. Then the scheduling program was run using the 4 different algorithms with the generated input files. The aim of this experiment was to use the same burst values across all algorithms to see which one had the smallest waiting time. This could not be achieved in previous experiments because even though the same avg and min values were used for value generation, the values were not exactly the same.



```
int main(int argc, char *argv[]){

   int m = atoi(argv[1]);
   int line;
   char fileName[50];
   char fileNameFormatted[50];
   strcpy(fileName, argv[2]);
   FILE *fp;

   for(int i = 0; i < m; i++){
       line = 20 + rand() % 30;

       strcpy(fileNameFormatted, fileName);
       strcat(fileNameFormatted,"-");
       char fileNo[5];
       snprintf(fileNo,5,"%d",i + 1);
       strcat(fileNameFormatted,fileNo);
       strcat(fileNameFormatted,".txt");
       fp = fopen(fileNameFormatted,"w");

       for(int j = 0; j < line; j++){
           fprintf(fp,"%d ", 100 + (rand() % 1000));
           fprintf(fp,"%d\n", 100 + (rand() % 1000));

       }
   }
   return(0);
}
```

One example of the 10 input files for 10 threads:



```
986 877
1015 893
435 486
592 749
521 462
127 790
159 863
1026 640
526 272
836 311
468 667
529 882
630 962
223 167
235 1029
902 122
158 169
267 493
556 111
142 329
473 521
1019 884
637 298
```

The result of this experiment is presented in the following table.

Thread Index	FCFS(s)	SJF(s)	PRIO(s)	VRUNTIME(s)
1	5.643	3.303	0.126	3.091
2	6.102	11.361	1.500	4.635
3	7.934	2.710	2.886	4.833
4	7.382	3.801	4.405	5.929
5	8.436	6.208	6.134	7.703
6	4.882	3.913	6.137	5.161
7	3.546	5.115	6.958	5.662
8	8.761	9.615	11.373	9.017
9	6.217	8.111	11.544	9.153
10	3.566	5.455	10.881	6.865
Avg of all threads	6.247	5.959	6.196	6.205

Best algorithm for threads based on experiment 7:

1. PRIO
2. PRIO
3. SJF
4. SJF
5. PRIO
6. SJF
7. FCFS
8. FCFS
9. FCFS
10. FCFS

On average, the best algorithm for these input files is SJF.