Jonathan Song

CSCI 3753

Programming Assignment 3 Report

Abstract

The purpose of this lab was to investigate the efficiency of the Linux Scheduler and its various scheduling policies with respect to various process types and levels of system utilization. By creating a series of benchmarks that would evaluate the behavior of the various test cases, I was able to draw conclusions on the efficiency of each policy. I found that in general, SCHED_OTHER, the default, standard CFS, had the most efficient runtime across each process type, but also was the least efficient policy in terms of overhead efficiency for each process type. SCHED_FIFO (first in first out scheduling policy) seemed to be the more overhead efficient policy for each process type, as it had the least context switches. However, SCHED_FIFO had less efficient runtimes than the SCHED_OTHER policy. The SCHED_RR (round robin policy) seemed to fall in between SCHED_OTHER and SCHED_FIFO in terms of efficiency in each case. From this data, it does not seem that there is a single clear winner among the scheduling policies in terms of runtime and overhead efficiency.

Introduction

This lab focuses on testing the Linux Scheduler and investigating how to best use the CPU with efficiency. The scheduler is a very valuable part of the OS that manages resources and efficiency uses the CPU to complete jobs. The Completely Fair Scheduler (CFS) utilized by the Linux Scheduler supports the SCHED_OTHER scheduling policy and the Real Time (RT) class provides the SCHED_RR and SCHED_FIFO scheduling policies. Using the current implementation of the Completely Fair Scheduler (CFS), I tested a variety of Linux scheduling policies with SCHED_OTHER, SCHED_FIFO, and SCHED_RR. These policies were tested with combinations of process types of CPU bound, I/O bound, and Mixed and combinations of levels of system utilization. Each test was performed on the Oracle VM with Ubuntu (64 bit).

Method

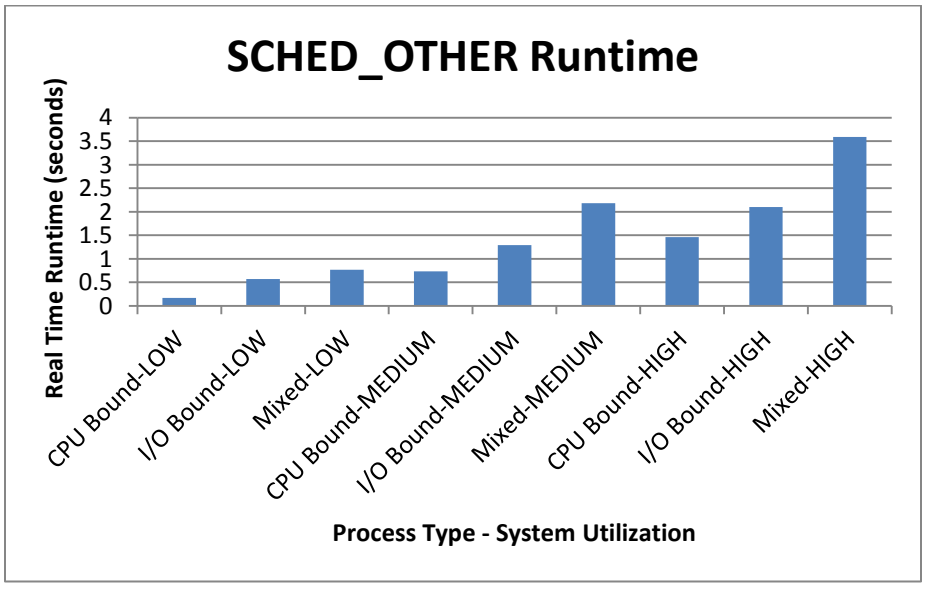
The code provided is able to test each of the 27 possible test cases for the Linux Scheduling policies combined with the various process types and levels of system utilization. I adapted code to test for the

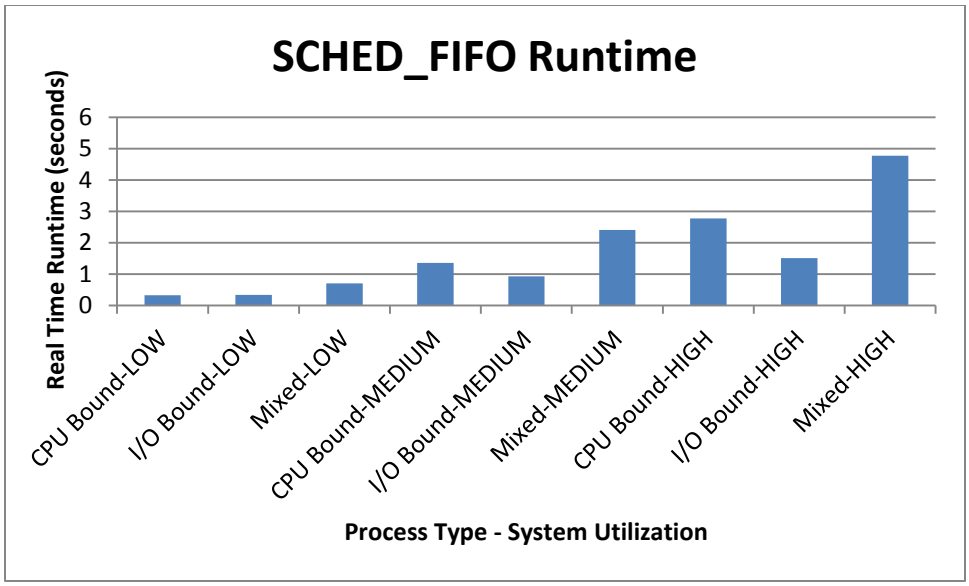
separate cases with respect to the process types based on some of the included code. For example, I used the code from pi.c to test for the CPU bound processes. The code for cpu.c was also modified so that it would accept the Linux Scheduling policies and system utilization as arguments. The code for io.c was adapted from rw.c to copy N bytes in blocks of K bytes from input file to an output file as an example of an I/O bound process. The code for mixed.c was a combination of both pi.c and rw.c, as it needed to test for a combination of I/O and CPU bound processes. To prevent ambiguity of data, I pre-set the number of simultaneous processes as the maximum of each particular level of system utilization. For example, Low system utilization was defined as 10, Medium system utilization was defined as 50, and High system utilization was defined as 100. Each run of the test case was controlled by the testscript, which runs each test several times so that the data may later be averaged and provide more meaningful data.

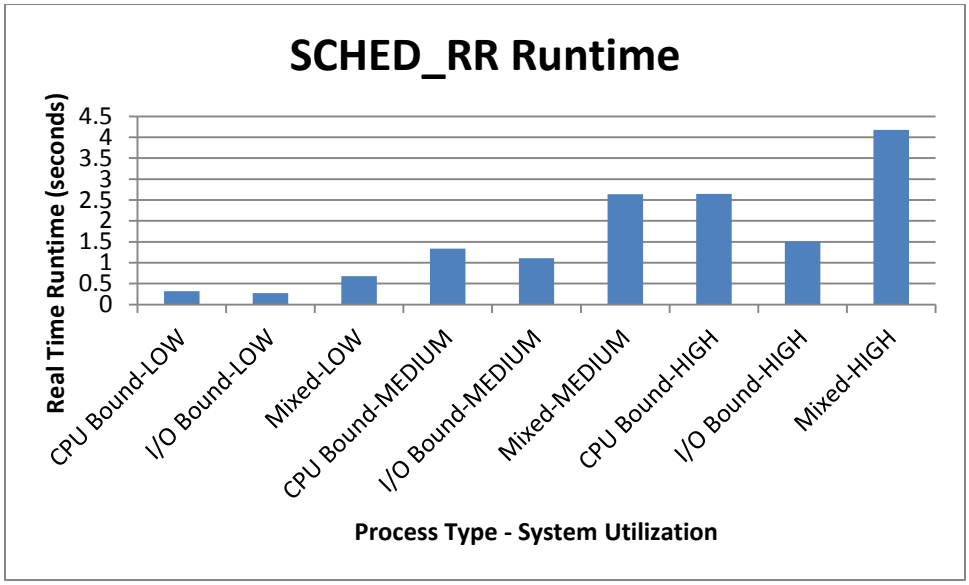
Results

The full table of results in available within Appendix A.

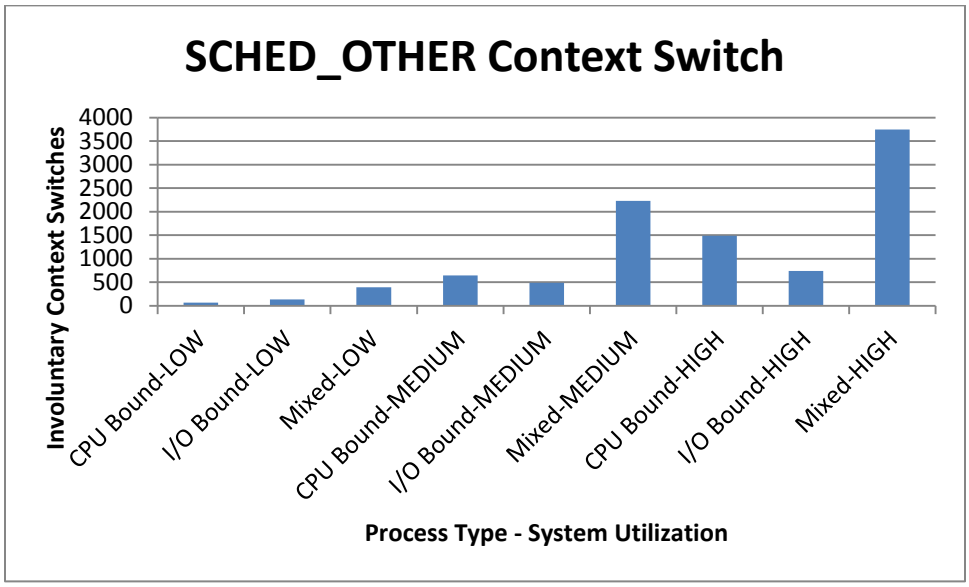
In terms on runtime, SCHED_OTHER seems to be the more efficient Linux Scheduling Policy as it generally seems to have lower real time runtimes than SCHED_FIFO and SCHED_RR, which seem to take roughly the same running times. Also, the I/O bound processes seem to run more quickly than CPU bound or Mixed processes.

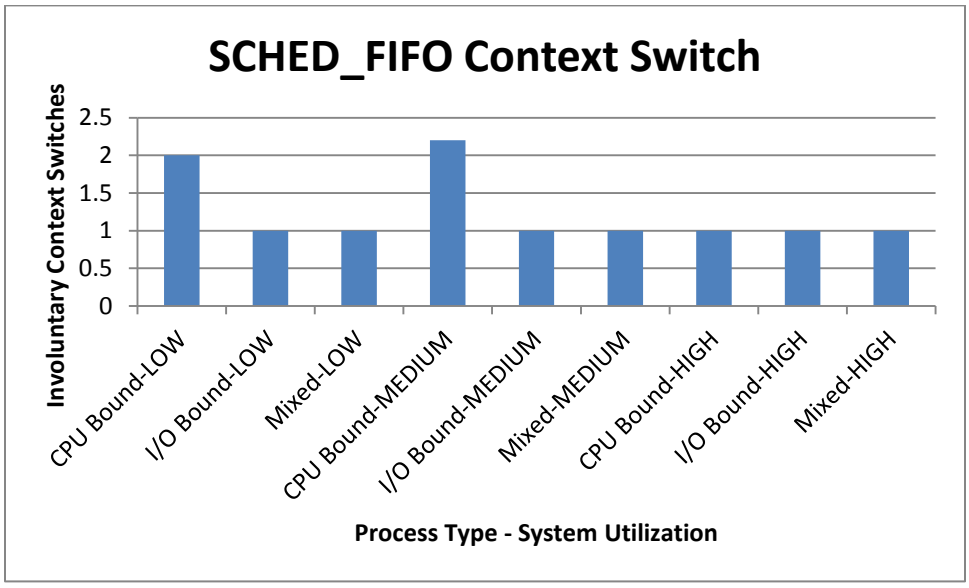


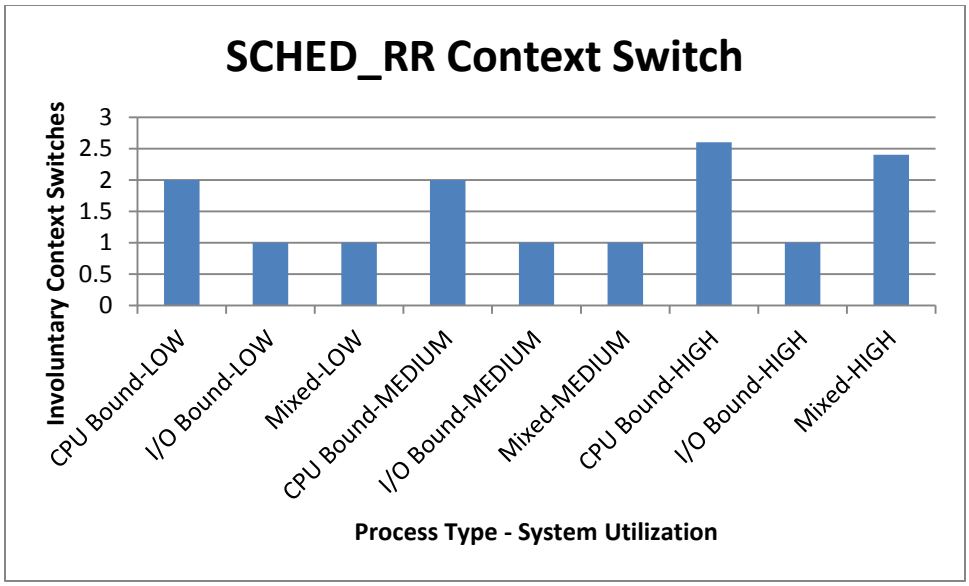

SCHED_FIFO Runtime
Real Time Runtime (seconds)
0
1
2
3
4
5
6
CPU Bound-LOW
I/O Bound-LOW
Mixed-LOW
CPU Bound-MEDIUM
I/O Bound-MEDIUM
Mixed-MEDIUM
CPU Bound-HIGH
I/O Bound-HIGH
Mixed-HIGH
Process Type - System Utilization


SCHED_RR Runtime
Real Time Runtime (seconds)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
CPU Bound-LOW
I/O Bound-LOW
Mixed-LOW
CPU Bound-MEDIUM
I/O Bound-MEDIUM
Mixed-MEDIUM
CPU Bound-HIGH
I/O Bound-HIGH
Mixed-HIGH
Process Type - System Utilization

In terms of context switches, SCHED_OTHER seems to make many more involuntary context switches than SCHED_FIFO or SCHED_RR. In SCHED_OTHER, the majority of context switches occur with mixed process types. SCHED_FIFO makes most context switches in CPU bound processes and SCHED_RR makes most context switches with CPU and Mixed processes.







Analysis

When using the CFS implementation of the Linux kernel, it seems that SCHED_OTHER is the best Linux Scheduling policy for CPU bound processes to use in terms of performing several jobs, as it seems to have the lowest real runtimes. However, SCHED_OTHER has many more context switches than the other scheduling policies. In this case, it seems that SCHED_FIFO is the better scheduling policy for most efficient overhead for CPU bound processes. SCHED_FIFO also seems to be the better scheduling policy for execution time and overhead efficiency for the I/O bound processes, as it generally seems to have lower runtimes and context switches. SCHED_OTHER seems to provide the most efficient runtime for mixed processes, while SCHED_FIFO seems to have the more efficient overhead for the mixed processes. SCHED_RR seems to be similarly efficient to SCHED_FIFO, but SCHED_FIFO mostly seems to be just a bit more efficient than SCHED_RR. In general, SCHED_OTHER has better runtimes, but much more context switches and a higher overhead. SCHED_FIFO has higher runtimes and better overhead, and SCHED_RR falls close to the results of SCHED_FIFO, but mostly falls just short of SCHED_FIFO efficiency.

Conclusion

Based on the results of the lab, it can be concluded that SCHED_OTHER is the better Linux Scheduling Policy to use in terms of most efficient runtime and should be used when one needs to run several jobs that need to finish quickly. However, SCHED_OTHER has the least efficient overhead as it seems to consistently make the most context switches due to its implementation of time sharing between the processes. Also, SCHED_OTHER's runtime and overhead mostly increases from CPU to I/O to mixed processes. The SCHED_FIFO scheduling policy seems to have better runtimes for I/O and mixed processes than for CPU bound processes and generally has the least context switches than the other scheduling policies. The SCHED_RR scheduling policy seems to fall in between SCHED_OTHER and SCHED_FIFO in terms of runtime and overhead efficiency. Based on these conclusions, it does not seem

that there is a clear winner for each process type, as they often seem to provide similar results or sacrifice overhead efficiency for runtime efficiency or vice versa.

References

Appendix A – Raw Data

Scheduling Policy	Process Type	System Utilization	Trial	Real Time	User Time	Kernel Time	CPU Utilization (%)
SCHED_OTHER	CPU Bound	Low	1	0.18	0.6	0	333
			2	0.16	0.58	0.01	357
			3	0.15	0.54	0.01	361
			4	0.16	0.57	0.02	329
			5	0.18	0.55	0.02	361
			average	0.166	0.568	0.012	348.2
	I/O Bound		1	0.56	0.01	0.36	67
			2	0.53	0.01	0.35	65
			3	0.6	0	0.34	55
			4	0.57	0.02	0.3	53
			5	0.6	0.05	0.41	70
			average	0.572	0.018	0.352	62
	Mixed		1	0.8	0.71	0.34	130
			2	0.78	0.63	0.4	132
			3	0.78	0.7	0.29	120
			4	0.68	0.64	0.22	139
			5	0.8	0.72	0.42	130
			average	0.768	0.68	0.334	130.2
	CPU Bound	Medium	1	0.75	2.72	0.05	367
			2	0.72	2.61	0.04	376
			3	0.7	2.65	0.03	373
			4	0.73	2.66	0.04	370
			5	0.78	2.72	0.08	355
			average	0.736	2.672	0.048	368.2
	I/O Bound		1	1.28	0.03	0.91	73
			2	1.14	0.04	0.88	78
			3	1.26	0.03	1.22	81
			4	1.33	0.04	1.02	84
			5	1.45	0.05	0.99	78
			average	1.292	0.038	1.004	78.8
	Mixed		1	2.23	3.15	0.9	180
			2	2.21	2.92	1.01	182
			3	2.25	2.97	1.15	179

			4	1.99	2.95	1.3	188
			5	2.22	3.03	0.85	192
			average	2.18	3.004	1.042	184.2
	CPU Bound	High	1	1.46	5.26	0.14	371
			2	1.41	5.48	0.11	376
			3	1.53	5.21	0.13	363
			4	1.44	5.25	0.1	372
			5	1.45	5.3	0.16	375
			average	1.458	5.3	0.128	371.4
	I/O Bound		1	2.07	0.05	1.66	80
			2	2.15	0.01	1.73	82
			3	1.98	0.02	1.52	81
			4	2	0.07	2.55	93
			5	2.31	0.09	1.53	79
			average	2.102	0.048	1.798	83
	Mixed		1	3.93	5.93	2.1	204
			2	3.43	5.72	1.9	222
			3	3.49	5.48	1.84	209
			4	3.53	5.57	2.06	215
			5	3.57	5.54	1.89	210
			average	3.59	5.648	1.958	212
SCHED_FIFO	CPU Bound	Low	1	0.35	0.32	0	93
			2	0.32	0.3	0	92
			3	0.33	0.29	0	90
			4	0.32	0.3	0	93
			5	0.32	0.29	0	91
			average	0.328	0.3	0	91.8
	I/O Bound		1	0.28	0.01	0.06	25
			2	0.38	0.01	0.06	20
			3	0.23	0	0.04	24
			4	0.27	0	0.05	19
			5	0.51	0	0.04	17
			average	0.334	0.004	0.05	21
	Mixed		1	0.83	0.53	0.46	119
			2	0.77	0.3	0.2	65
			3	0.58	0.32	0.08	75
			4	0.75	0.45	0.15	65
			5	0.6	0.32	0.08	60
			average	0.706	0.384	0.194	76.8
	CPU	Medium	1	1.01	1.77	0.01	176

	Bound						
			2	1.57	1.4	0.02	90
			3	1.09	2.05	0.02	187
			4	1.59	1.44	0.01	91
			5	1.5	1.58	0.02	91
			average	1.352	1.648	0.016	127
	I/O Bound		1	0.9	0.03	0.26	32
			2	0.76	0.04	0.16	26
			3	0.81	0.04	0.19	29
			4	1.1	0.03	0.38	38
			5	1.04	0.04	0.17	27
			average	0.922	0.036	0.232	30.4
	Mixed		1	2.61	2.28	1.5	145
			2	2.45	2.6	0.3	75
			3	2	1.6	1.45	218
			4	2.45	1.5	0.4	95
			5	2.5	1.75	0.6	140
			average	2.402	1.946	0.85	134.6
	CPU Bound	High	1	3.13	3.01	0.04	97
			2	3.18	2.86	0.04	91
			3	1.8	4.68	0.04	262
			4	2.4	3.92	0.06	167
			5	3.34	2.95	0.09	92
			average	2.77	3.484	0.054	141.8
	I/O Bound		1	1.32	0.1	0.34	33
			2	1.44	0.1	0.34	32
			3	1.3	0.09	0.32	32
			4	1.61	0.07	0.79	45
			5	1.86	0.06	1.13	53
			average	1.506	0.084	0.584	39
	Mixed		1	5.1	3.3	1.49	95
			2	4.6	4.09	2.2	135
			3	4.25	4.45	2.2	157
			4	5.2	3.14	1.3	85
			5	4.7	3.7	1.5	115
			average	4.77	3.736	1.738	117.4
SCHED_RR	CPU Bound	Low	1	0.34	0.32	0	92
			2	0.33	0.3	0	93
			3	0.31	0.3	0	91
			4	0.3	0.28	0	92

			5	0.31	0.29	0	91
			average	0.318	0.298	0	91.8
	I/O Bound		1	0.25	0	0.08	31
			2	0.25	0	0.11	47
			3	0.3	0.01	0.04	28
			4	0.3	0.02	0.03	15
			5	0.26	0	0.08	23
			average	0.272	0.006	0.068	28.8
	Mixed		1	0.62	0.33	0.07	67
			2	0.77	0.34	0.13	61
			3	0.75	0.29	0.17	62
			4	0.68	0.31	0.12	60
			5	0.59	0.3	0.13	68
			average	0.682	0.314	0.124	63.6
	CPU Bound	Medium	1	1.61	1.47	0.01	91
			2	1.59	1.45	0.01	92
			3	0.92	2.25	0.01	248
			4	1.35	1.52	0.06	114
			5	1.22	1.83	0.04	153
			average	1.338	1.704	0.026	139.6
	I/O Bound		1	0.97	0.05	0.2	25
			2	0.99	0.03	0.44	49
			3	1.66	0.04	0.38	28
			4	1.03	0.07	0.46	23
			5	0.89	0.04	0.16	45
			average	1.108	0.046	0.328	34
	Mixed		1	2.68	1.63	0.8	90
			2	2.69	1.55	0.33	68
			3	2.41	2.55	1.33	160
			4	2.55	1.45	0.45	72
			5	2.87	2.19	1.3	85
			average	2.64	1.874	0.842	95
	CPU Bound	High	1	3.05	3.06	0.03	101
			2	3.16	2.86	0.03	91
			3	2.02	4.68	0.06	265
			4	2.98	3.58	0.05	171
			5	2.02	4.85	0.11	287
			average	2.646	3.806	0.056	183
	I/O Bound		1	1.59	0.08	0.37	28
			2	1.54	0.1	0.36	30

			3	1.67	0.14	0.76	50
			4	1.38	0.08	0.38	75
			5	1.35	0.07	1.15	30
			average	1.506	0.094	0.604	42.6
	Mixed		1	4.04	4.66	2	165
			2	4.25	4.72	2.12	162
			3	4	4.78	2.25	170
			4	4.09	4.62	1.95	158
			5	4.5	4.84	1.7	169
			average	4.176	4.724	2.004	164.8

Scheduling Policy	**Process Type**	**System Utilization**	**Trial**	**Involuntary Context Switches**	**Voluntary Context Switches**
SCHED_OTHER	CPU Bound	Low	1	70	21
			2	66	21
			3	55	20
			4	58	22
			5	73	27
			average	64.4	22.2
	I/O Bound		1	160	3750
			2	140	3623
			3	92	3175
			4	137	3638
			5	137	3790
			average	133.2	3595.2
	Mixed		1	542	3691
			2	318	3777
			3	638	3528
			4	233	3174
			5	223	3305
			average	390.8	3495
	CPU Bound	Medium	1	622	100
			2	666	102
			3	611	99
			4	663	97
			5	653	100
			average	643	99.6
	I/O Bound		1	551	18318
			2	531	17640
			3	461	17399
			4	495	18076

			5	405	17963
			average	488.6	17879.2
	Mixed		1	2428	18137
			2	2277	18541
			3	2158	17658
			4	2104	17404
			5	2170	18192
			average	2227.4	17986.4
	CPU Bound	High	1	1355	199
			2	1377	197
			3	1475	210
			4	1515	198
			5	1717	201
			average	1487.8	201
	I/O Bound		1	767	36495
			2	786	36943
			3	729	34540
			4	1081	33010
			5	338	35937
			average	740.2	35385
	Mixed		1	4233	33800
			2	4159	34472
			3	3572	35119
			4	3887	35133
			5	2888	36063
			average	3747.8	34917.4
SCHED_FIFO	CPU Bound	Low	1	2	14
			2	2	14
			3	2	16
			4	2	17
			5	2	14
			average	2	15
	I/O Bound		1	1	2015
			2	1	3515
			3	1	2015
			4	1	2012
			5	1	2013
			average	1	2314
	Mixed		1	1	2901
			2	1	2907
			3	1	2012
			4	1	3912
			5	1	2084
			average	1	2763.2

	CPU Bound	Medium	1	2	55
			2	2	54
			3	2	57
			4	2	56
			5	3	53
			average	2.2	55
	I/O Bound		1	1	11360
			2	1	10182
			3	1	11222
			4	1	14775
			5	1	14915
			average	1	12490.8
	Mixed		1	1	15206
			2	1	10184
			3	1	17580
			4	1	15028
			5	1	10817
			average	1	13763
	CPU Bound	High	1	1	107
			2	2	104
			3	2	106
			4	2	105
			5	1	107
			average	1	105.8
	I/O Bound		1	1	23333
			2	1	28588
			3	1	24333
			4	1	30600
			5	1	30216
			average	1	27414
	Mixed		1	1	29890
			2	1	30750
			3	1	31276
			4	1	30349
			5	1	30600
			average	1	30573
SCHED_RR	CPU Bound	Low	1	2	15
			2	2	14
			3	2	16
			4	2	13
			5	2	14
			average	2	14.4
	I/O Bound		1	1	2045
			2	1	2021

			3	1	2012
			4	1	2027
			5	1	2049
			average	1	2030.8
	Mixed		1	1	2055
			2	1	3002
			3	1	3041
			4	1	2909
			5	1	3121
			average	1	2825.6
	CPU Bound	Medium	1	2	55
			2	2	54
			3	2	65
			4	2	57
			5	2	55
			average	2	57.2
	I/O Bound		1	1	15027
			2	1	14040
			3	1	14655
			4	1	13741
			5	1	14466
			average	1	14385.8
	Mixed		1	1	15522
			2	1	16266
			3	1	14120
			4	1	10608
			5	1	18375
			average	1	14978.2
	CPU Bound	High	1	2	105
			2	2	104
			3	3	108
			4	4	103
			5	2	107
			average	2.6	105.4
	I/O Bound		1	1	32760
			2	1	28294
			3	1	32592
			4	1	30495
			5	1	23578
			average	1	29543.8
	Mixed		1	1	30617
			2	2	30881
			3	4	30912
			4	3	30720
			5	2	30041

			average	2.4	30634.2

Appendix B – Brief Description of Source Files

cpu.c –Code based off of pi.c to calculate pi as an example of a CPU bound process. Also accepts Linux Scheduling policy and system utilization as inputs.

io.c – Code bases off of rw.c to copy N bytes in blocks of K bytes from an input file to an output file as an example of an I/O bound process. Also accepts Linux Scheduling policy and system utilization as inputs.

mixed.c – Code based off of both pi.c and rw.c to test both CPU and I/O bound processes. Also accepts Linux Scheduling policy and system utilization as inputs.

README – Provides information on files and how to run them.

testscript - Bash script to run the 27 test cases

Makefile – Compiles the code

tests_output – stores output data for test cases