

```
1 CS 460 Scheduling Lab
2
3 Shutdown VB and change the System to use 2 CPUs.
4
5 Open terminator!
6
7 wget http://zeus.cs.pacificu.edu/chadd/cs460s18/SchedLab.tar.gz
8
9 tar xzf SchedLab.tar.gz
10
11 cd CS460_SchedulingLab
12
13 make
14
15 This produces a number of executables. We will only use some
16 of these executables today.
17
18 Split the terminator window horizontally.
19
20 Split the bottom window vertically.
21
22
23 CONFIGURE TOP
24
25 In the top window:
26
27 taskset -c 0 top -u punetid
28
29 V
30 H
31 f
32 <arrow down to P = Last Used CPU>
33 <space>
34 <arrow up to %CPU>
35 s
36 q
37 s .3
38
39
40 RUN THE EXECUTABLES:
41
42 In either small window:
43
44 Try out a few of the executables. Note how much work each executable
45 reports it has done and how many voluntary and involuntary context
46 switches occur.
47
```

```

48 ./sleeper 20
49
50 Work: _____ VOL CS: _____ IVOL CS: _____
51
52 ./CPU 20
53
54 Work: _____ VOL CS: _____ IVOL CS: _____
55
56 ./IO 20
57
58 Work: _____ VOL CS: _____ IVOL CS: _____
59
60 Each of the previous executables takes a command line argument that
61 is the runtime in seconds for the process. In the above examples,
62 each process should run for very close to 20 seconds.
63
64 Each executable reports the amount of work done and the number
65 of voluntary and involuntary context switches done by that process.
66
67 sleeper just continually calls sleep(1) until the runtime is expired.
68
69 CPU runs a for loop and does some calculations until the runtime
70 is expired.
71
72 IO runs a for loop and prints data to stderr until the runtime is
73 expired.
74
75
76 Group the two smaller windows.
77
78 Box menu | New Group ... <enter>
79
80 Box menu | select group
81
82
83 Alt-G <send input to all windows in a group>
84
85 ./CPU 20 # should appear in both small windows
86
87
88 ./CPU 20
89
90 Work: _____ VOL CS: _____ IVOL CS: _____
91
92
93 ./CPU 20
94

```

95 Work: _____ VOL CS: _____ IVOL CS: _____
96
97
98 How does the amount of work compare for the two processes?
99
100 How does the amount of work compare against a single ./CPU 20 process?
101
102
103 Let's restrict both processes to the same CPU:
104
105 taskset -c 1 ./CPU 20 # should appear in both small windows
106
107 taskset -c 1 ./CPU 20
108
109 Work: _____ VOL CS: _____ IVOL CS: _____
110
111
112 taskset -c 1 ./CPU 20
113
114 Work: _____ VOL CS: _____ IVOL CS: _____
115
116
117 How does the amount of work compare for the two processes?
118
119 How does the amount of work compare against two CPU processes running
120 without being restricted to a CPU?
121
122
123 Let's restrict both processes to different CPUs:
124
125 Alt-0 # Alt-0h ungroup input
126
127 Type in each command below but don't press enter.
128
129 Alt-G # regroup input
130
131 Press enter
132
133 taskset -c 0 ./CPU 20
134
135 Work: _____ VOL CS: _____ IVOL CS: _____
136
137
138 taskset -c 1 ./CPU 20
139
140 Work: _____ VOL CS: _____ IVOL CS: _____
141

142
143 How does the amount of work compare for the two processes?
144
145 How does the amount of work compare against two CPU processes running
146 without being restricted to a CPU?
147
148
149
150 Let's add a sleeper process to the same CPU:
151
152 Alt-0 # ungroup input
153
154 Type in each command below but don't press enter.
155
156 Alt-G # regroup input
157
158 Press enter
159
160 taskset -c 1 ./CPU 20
161
162 Work: _____ VOL CS: _____ IVOL CS: _____
163
164
165 taskset -c 1 ./sleeper 20
166
167 Work: _____ VOL CS: _____ IVOL CS: _____
168
169
170 How does the amount of work compare for the two processes?
171
172 How does the amount of work compare against two CPU processes running
173 without being restricted to a CPU?
174
175
176
177 Let's add an IO process to the same CPU:
178
179 Alt-0 # ungroup input
180
181 Type in each command below but don't press enter.
182
183 Alt-G # regroup input
184
185 Press enter
186
187 taskset -c 1 ./CPU 20
188

189 Work: _____ VOL CS: _____ IVOL CS: _____

190

191

192 taskset -c 1 ./IO 20

193

194 Work: _____ VOL CS: _____ IVOL CS: _____

195

196

197 How does the amount of work compare for the two processes?

198

199 How does the amount of work compare against two CPU processes running
200 without being restricted to a CPU?

201

202

203

204

205 SCHEDULING

206

207 Linux has a number of scheduling algorithms available:

208

209 Real time processes:

210 SCHED_FIFO

211 SCHED_RR

212

213 Everything else:

214 SCHED_OTHER

215 SCHED_BATCH

216 SCHED_IDLE

217

218 Read the man page for sched to understand each algorithm.

219

220 Step 0:

221

222 schedTest launches 5 threads (via pthreads) and pins those threads
223 to core 1 and sets the scheduling policy based on the command line
224 argument given. R is RR and F is FIFO.

225

226 Each thread prints 10 messages containing the thread ID, a progress
227 number, and a time stamp.

228

229 At the end, schedTest prints out the number of voluntary and
230 involuntary context switches that occurred.

231

232 Note: all output happens just before a process terminates so as to not
233 generate any extra context switches via printf.

234

235 time sudo ./schedTest R

236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282

```
time sudo ./schedTest F
```

Does it seem like the scheduler is working correctly?

Justify your answer.

Step 1:

Note: RT scheduling priorities run from 1-99. 99 is highest priority

Note: all output happens just before a process terminates so as to not generate any extra context switches via printf.

schedTestFork launches argv[3] processes (via fork()) and pins those threads to core 1 and sets the scheduling policy based on argv[1]. R is RR and F is FIFO, B is BATCH, I is IDLE, O is OTHER.

Each thread prints 10 messages containing the thread ID, a progress number, and a time stamp.

argv[2] sets the priority of each process via:
(threadid % argv[2]) + 1 # threadid starts at 0 and increments by 1

As each process ends, the counts of voluntary and involuntary context switches are listed.

Run each of the following command individually.

Because schedTestFork changes its scheduling algorithm, you must run schedTestFork with root privileges.

```
time sudo ./schedTestFork R 1 2
```

```
time sudo ./schedTestFork R 98 2
```

```
time sudo ./schedTestFork F 1 2
```

```
time sudo ./schedTestFork F 98 2
```

Does it seem like the scheduler is working correctly?

Justify your answer.

283 Step 1.1:
284 In a split Terminator window, run the following with grouped input:
285
286 time sudo ./schedTestFork R 98 2
287 time ./CPU 20 work: _____
288
289 time (real): _____
290

291 Just in one terminal alone:
292 time taskset -c 1 ./CPU 10 work: _____
293
294 time (real): _____
295
296
297

298 time sudo ./schedTestFork R 98 10
299 time taskset -c 1 ./CPU 10 work: _____
300
301 time (real): _____
302
303

304 time sudo ./schedTestFork R 98 20
305 time taskset -c 1 ./CPU 10 work: _____
306
307 time (real): _____
308
309

310 What happens to the work for CPU?

311
312 What happens for the real time for CPU?

313
314
315 Start in 2 way terminator window to see how processes with different
316 scheduling algorithms interact.

317
318 The important behavior to watch for is what order the processes
319 complete in and how the processes interleave their execution.

320
321 time sudo ./schedTestFork R 1 5
322 time sudo ./schedTestFork R 1 5
323

324 How do the processes interleave within one execution of schedTestFork?

325
326
327
328 How do the processes interleave between the two schedTestFork
329 processes?

330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376

Do any processes have voluntary context switches?

```
time sudo ./schedTestFork R 1 5  
time sudo ./schedTestFork R 98 5
```

How do the processes interleave within one execution of schedTestFork?

How do the processes interleave between the two schedTestFork processes?

```
time sudo ./schedTestFork R 1 5  
time sudo ./schedTestFork I 1 5
```

How do the processes interleave within one execution of schedTestFork?

How do the processes interleave between the two schedTestFork processes?

```
time sudo ./schedTestFork R 1 5  
time sudo ./schedTestFork F 1 5
```

How do the processes interleave within one execution of schedTestFork?

How do the processes interleave between the two schedTestFork processes?

377
378
379
380
381
382
383
384
385
386
387
388
389
390

```
time sudo ./schedTestFork R 1 5  
time sudo ./schedTestFork B 1 5
```

How do the processes interleave within one execution of schedTestFork?

How do the processes interleave between the two schedTestFork processes?