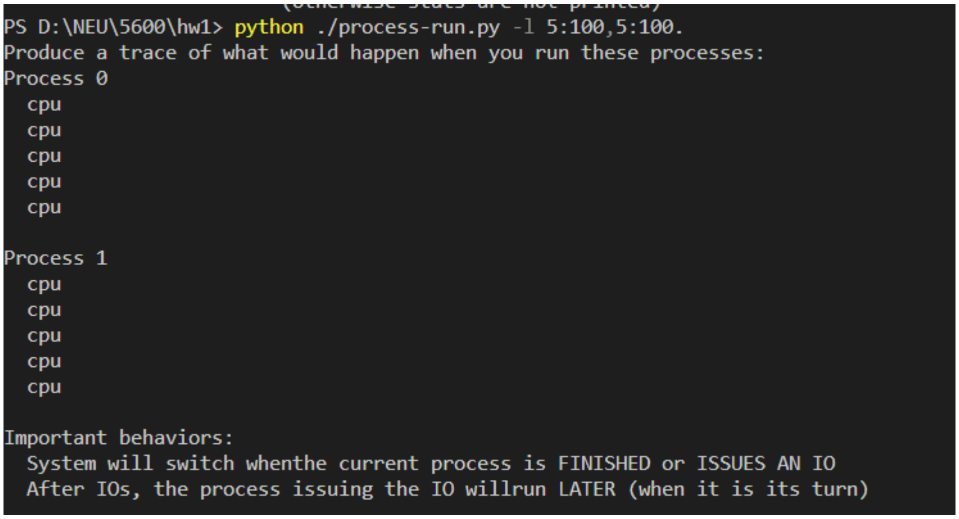
5600

Ziyun Chen

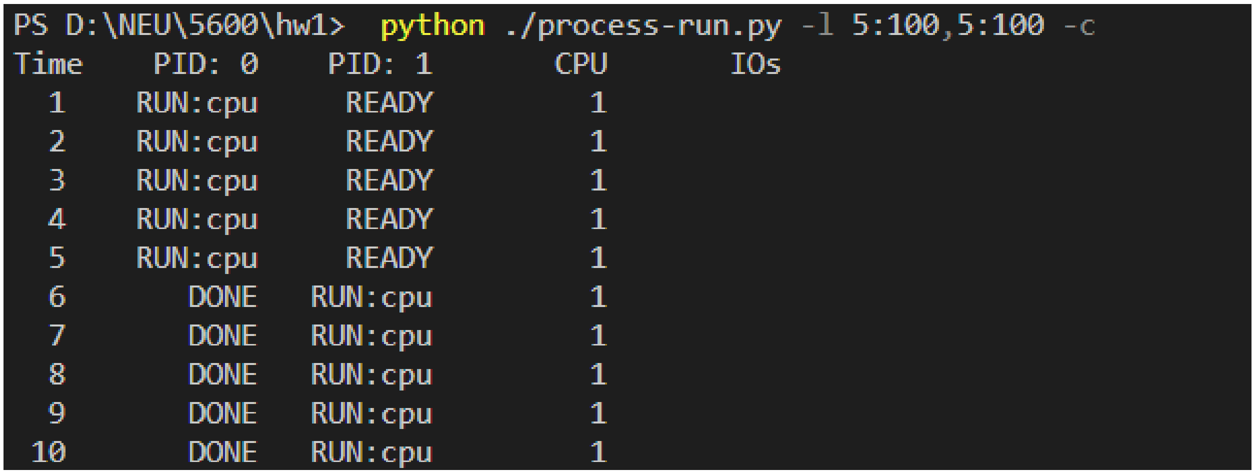
HW1 - Chapter 4

Q1: By running the process-run.py file using the flag ”./process-run.py -l 5:100,5:100”, I got the following result:

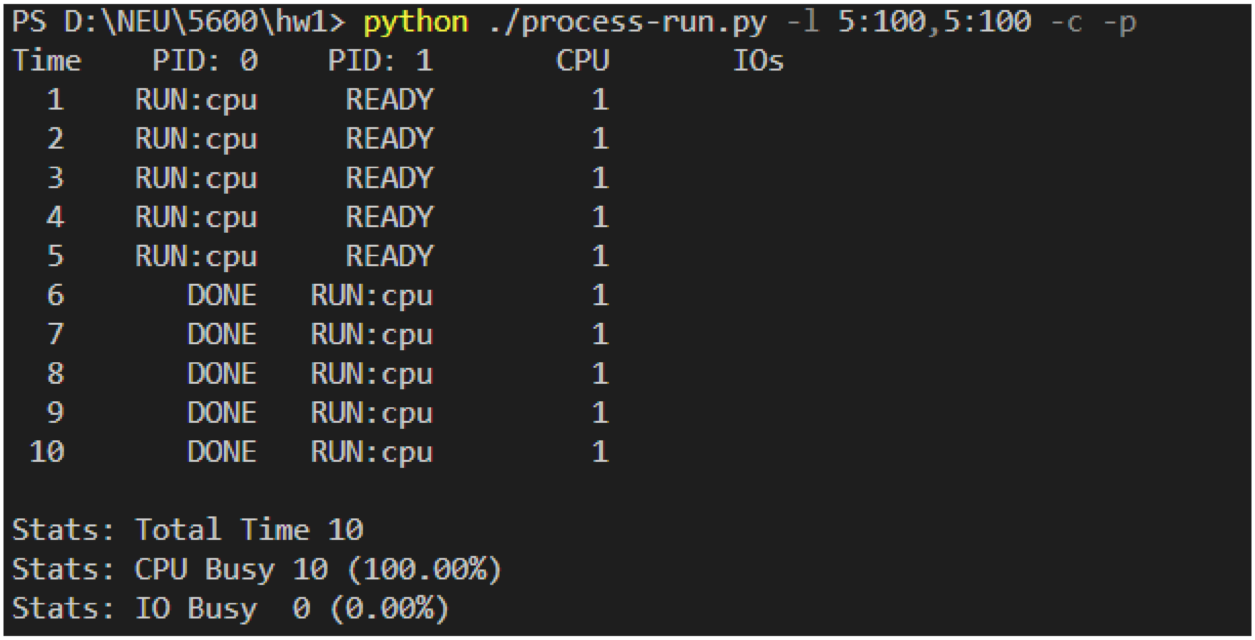


```
                              (otherwise stats are not printed)
PS D:\NEU\5600\hw1> python ./process-run.py -l 5:100,5:100.
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu
  cpu

Process 1
  cpu
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch whenthe current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO willrun LATER (when it is its turn)
```

From this it seems that the cpu is constantly being used. So, the percent of time the CPU is in use is 100%.



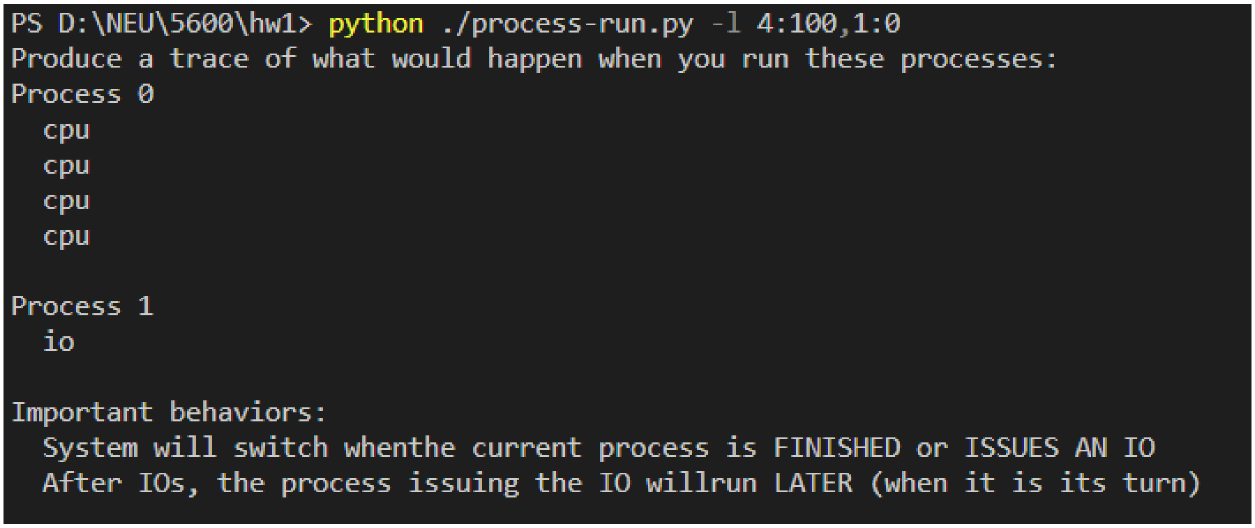
```
PS D:\NEU\5600\hw1>  python ./process-run.py -l 5:100,5:100 -c
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2     RUN:cpu      READY          1
  3     RUN:cpu      READY          1
  4     RUN:cpu      READY          1
  5     RUN:cpu      READY          1
  6        DONE    RUN:cpu          1
  7        DONE    RUN:cpu          1
  8        DONE    RUN:cpu          1
  9        DONE    RUN:cpu          1
 10        DONE    RUN:cpu          1
```



```
PS D:\NEU\5600\hw1> python ./process-run.py -l 5:100,5:100 -c -p
Time    PID: 0    PID: 1      CPU       IOs
  1    RUN:cpu     READY        1
  2    RUN:cpu     READY        1
  3    RUN:cpu     READY        1
  4    RUN:cpu     READY        1
  5    RUN:cpu     READY        1
  6       DONE   RUN:cpu        1
  7       DONE   RUN:cpu        1
  8       DONE   RUN:cpu        1
  9       DONE   RUN:cpu        1
 10       DONE   RUN:cpu        1

Stats: Total Time 10
Stats: CPU Busy 10 (100.00%)
Stats: IO Busy  0 (0.00%)
```

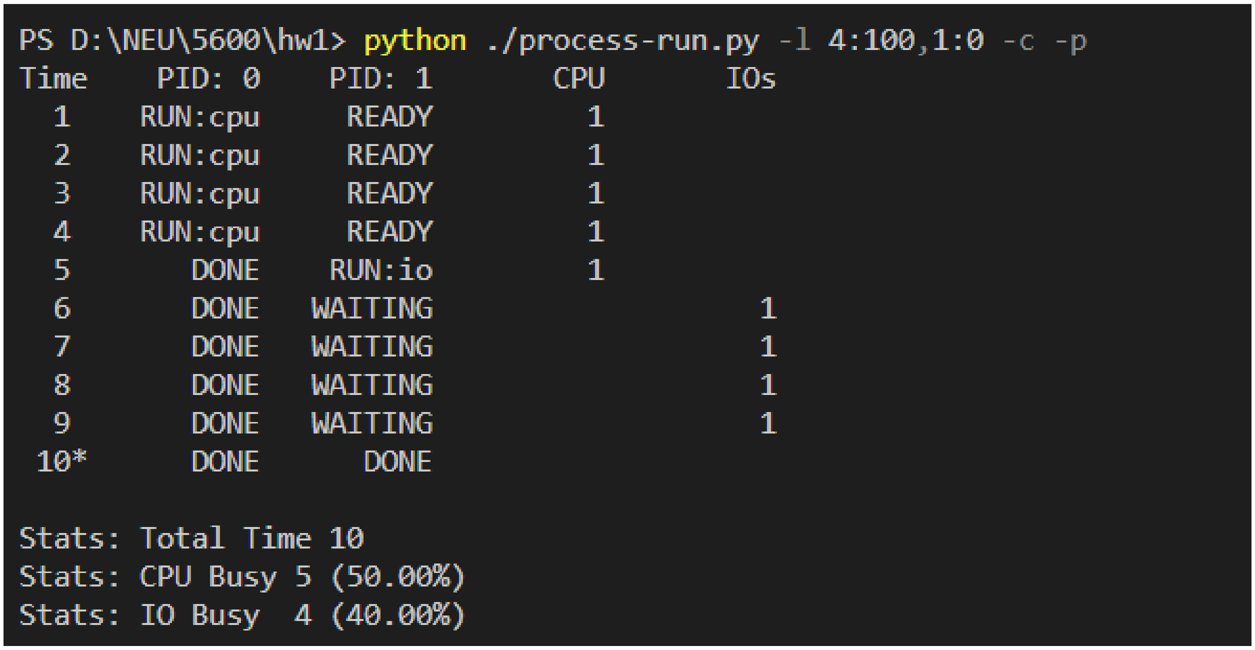
But adding -c and -p command, it proves that CPU is 100% in use.

Q2: When running with “./process-run.py -l 4:100,1:0”, I got this result:



```
PS D:\NEU\5600\hw1> python ./process-run.py -l 4:100,1:0
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu

Process 1
  io

Important behaviors:
  System will switch whenthe current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO willrun LATER (when it is its turn)
```

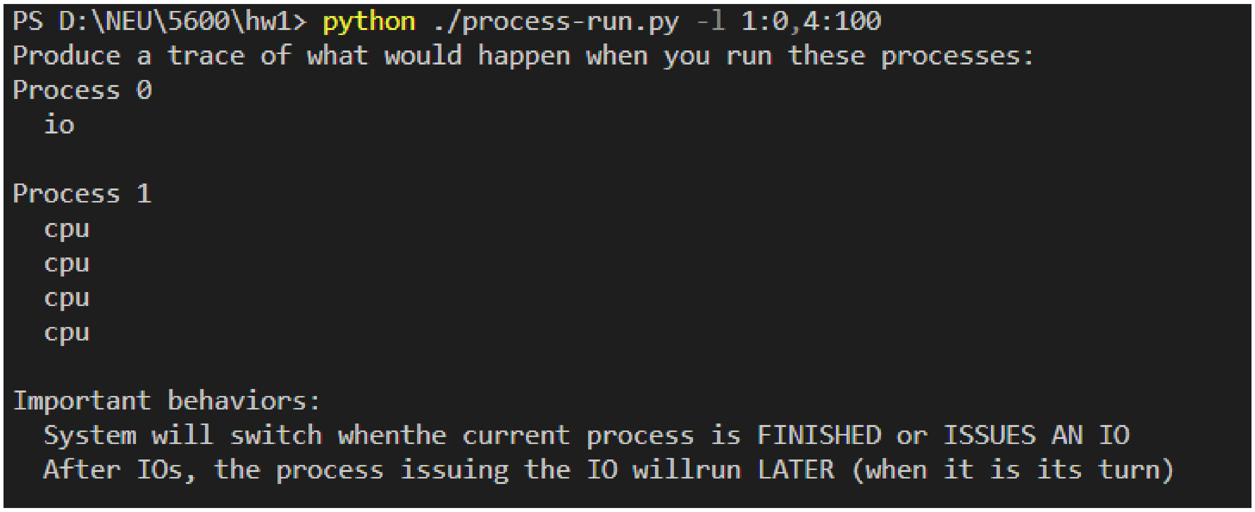
But after running it with -c and -p, it turns out:



```
PS D:\NEU\5600\hw1> python ./process-run.py -l 4:100,1:0 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2     RUN:cpu      READY          1
  3     RUN:cpu      READY          1
  4     RUN:cpu      READY          1
  5        DONE     RUN:io          1
  6        DONE    WAITING                     1
  7        DONE    WAITING                     1
  8        DONE    WAITING                     1
  9        DONE    WAITING                     1
 10*       DONE       DONE

Stats: Total Time 10
Stats: CPU Busy 5 (50.00%)
Stats: IO Busy  4 (40.00%)
```

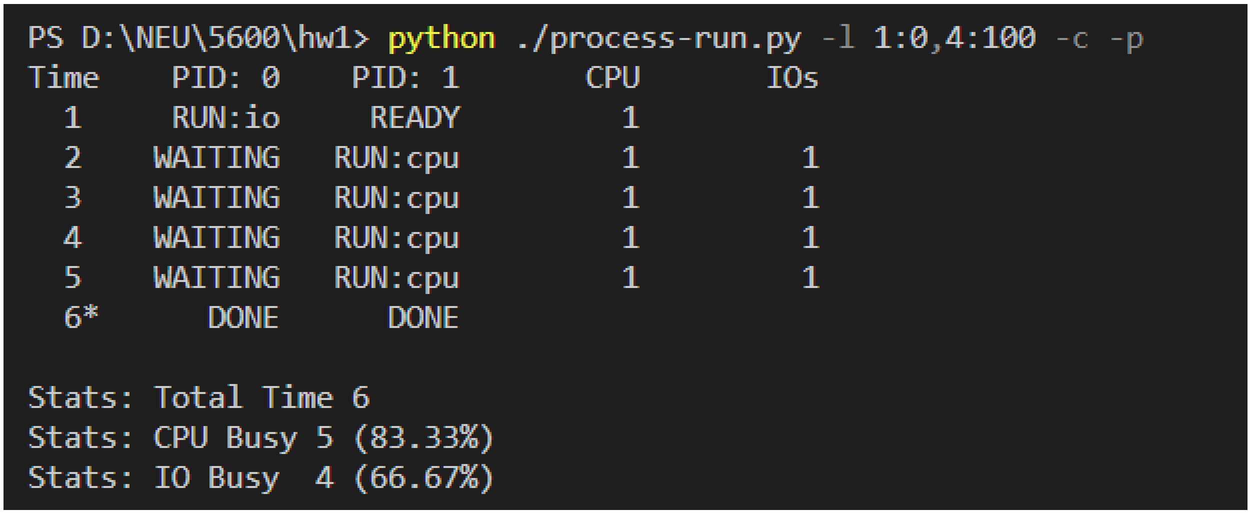
Process took 4, since it has 4 instructions, while process took 5, plus 1 final step. So the total time is 5+4+1 = 10.

Q3: After running the file with: ./process-run.py -l 1:0,4:100, I got:



```
PS D:\NEU\5600\hw1> python ./process-run.py -l 1:0,4:100
Produce a trace of what would happen when you run these processes:
Process 0
  io

Process 1
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch whenthe current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO willrun LATER (when it is its turn)
```

My guesses was: because from Q2 we know process 0 has 5 instructions in total. So, when process 0 started waiting for IO, it held, and switched to process 1. After process 1 was finished, it returned back to process 0. I assumed the total time should stay the same, which is 10. However, after I run the file with -c -p, I got:

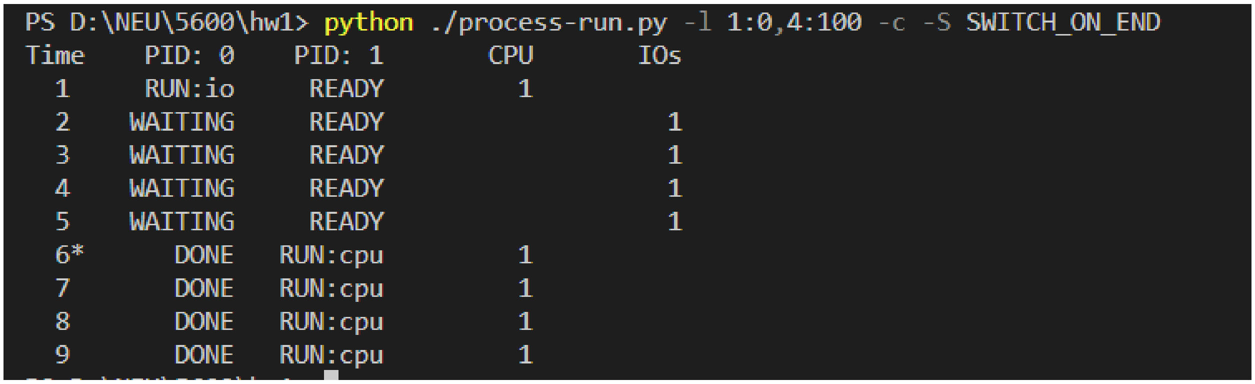


```
PS D:\NEU\5600\hw1> python ./process-run.py -l 1:0,4:100 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1      RUN:io      READY          1
  2     WAITING    RUN:cpu          1          1
  3     WAITING    RUN:cpu          1          1
  4     WAITING    RUN:cpu          1          1
  5     WAITING    RUN:cpu          1          1
  6*       DONE       DONE

Stats: Total Time 6
Stats: CPU Busy 5 (83.33%)
Stats: IO Busy  4 (66.67%)
```

The total time was reduced to 6. Once process 0 was waiting for IO, CPU was not used by process 0 anymore, so it started running process1. From 2 to 5, I/O and cpu were running at the same time. Which saved a lot of time. So, the order switch matters.

Q4:

Here is the result I got:

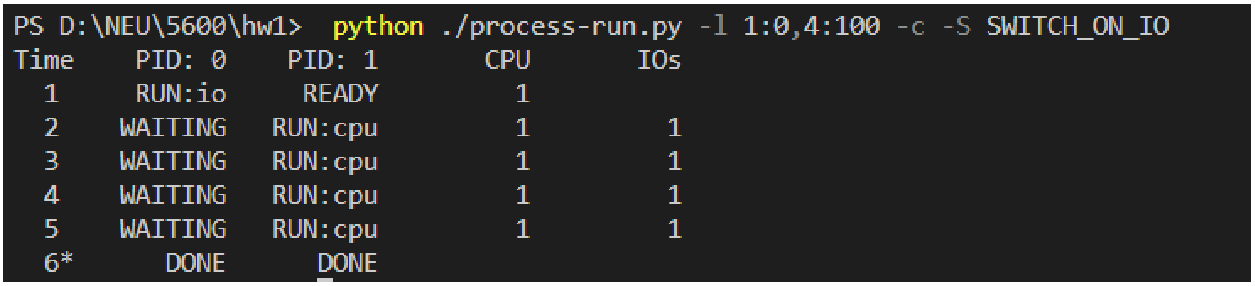


```
PS D:\NEU\5600\hw1> python ./process-run.py -l 1:0,4:100 -c -S SWITCH_ON_END
Time     PID: 0     PID: 1        CPU        IOs
  1      RUN:io      READY          1
  2     WAITING      READY                     1
  3     WAITING      READY                     1
  4     WAITING      READY                     1
  5     WAITING      READY                     1
  6*       DONE    RUN:cpu          1
  7        DONE    RUN:cpu          1
  8        DONE    RUN:cpu          1
  9        DONE    RUN:cpu          1
```

Process 1 didn’t start until process 0 was finished, even when process 0 was not using the cpu.

Q5:

Here is the result I got:

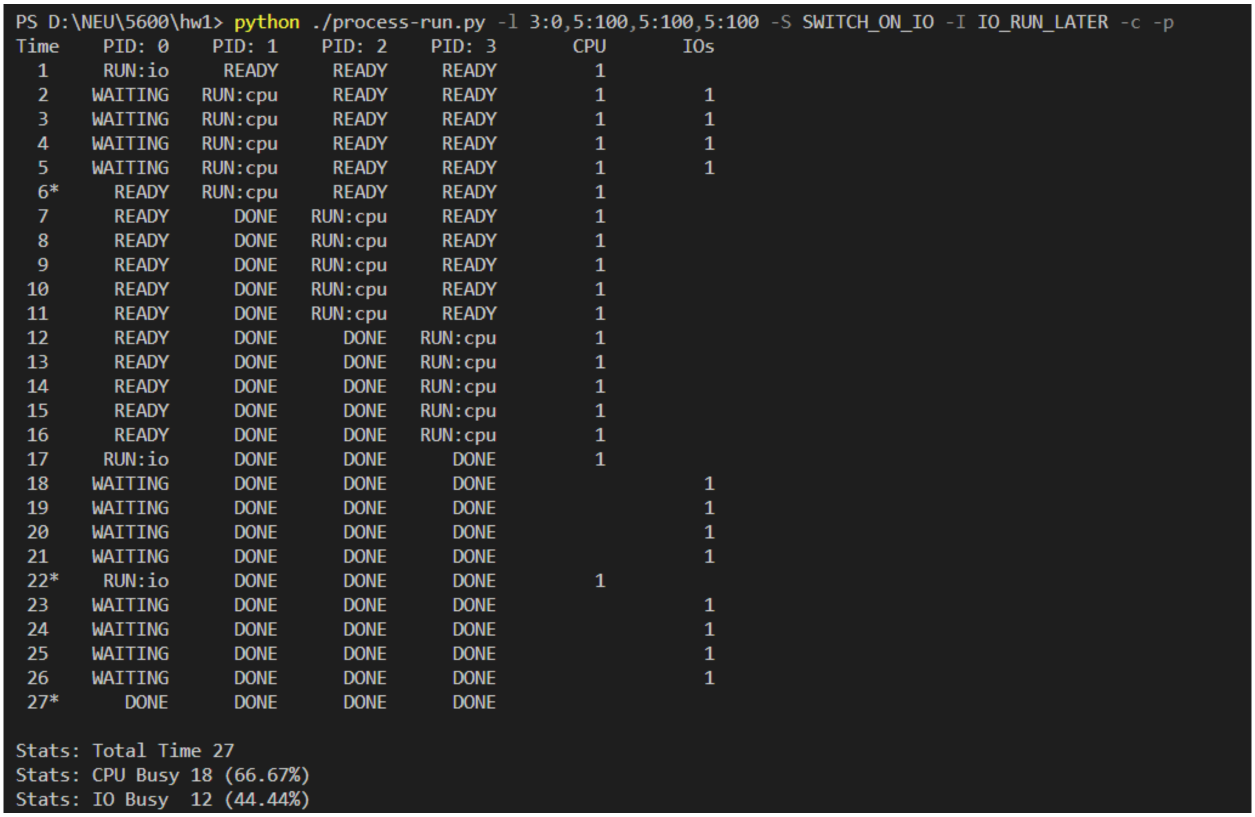


```
PS D:\NEU\5600\hw1>  python ./process-run.py -l 1:0,4:100 -c -S SWITCH_ON_IO
Time     PID: 0     PID: 1        CPU        IOs
  1      RUN:io      READY          1
  2     WAITING    RUN:cpu          1          1
  3     WAITING    RUN:cpu          1          1
  4     WAITING    RUN:cpu          1          1
  5     WAITING    RUN:cpu          1          1
  6*       DONE       DONE
```

We achieved the same result as we did from Q3. Process 1 started as soon as process 0 started waiting.

Q6:

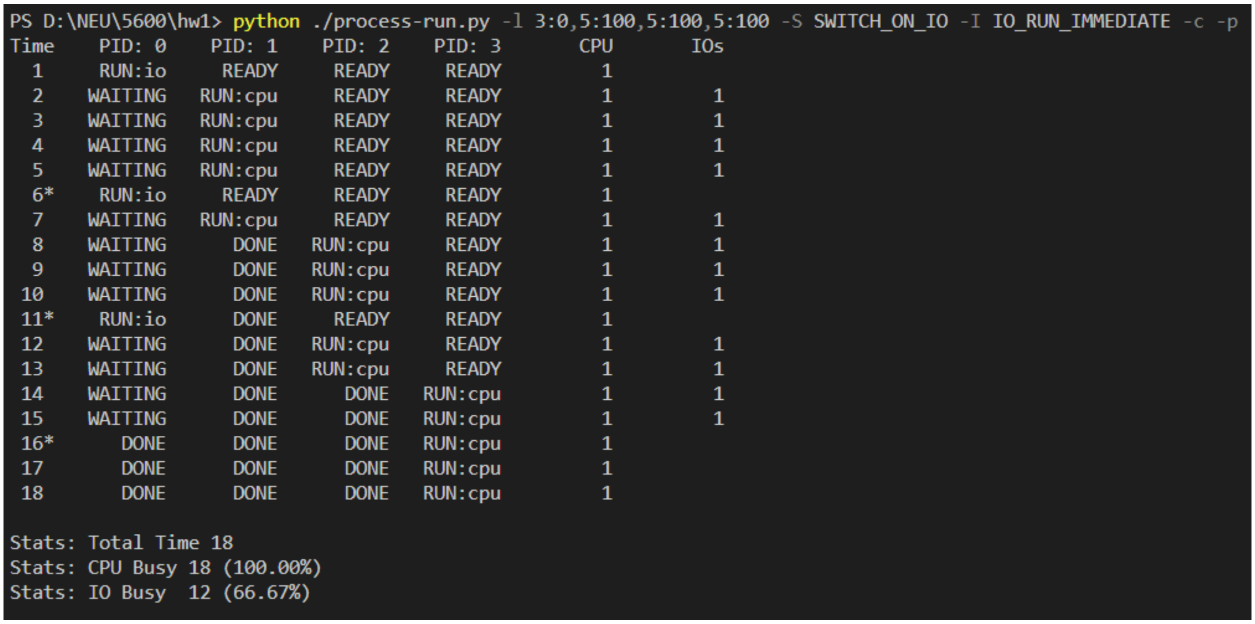
Here is the result I got,



```
PS D:\NEU\5600\hw1> python ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c -p
Time     PID: 0     PID: 1     PID: 2     PID: 3        CPU        IOs
  1      RUN:io      READY      READY      READY          1
  2     WAITING    RUN:cpu      READY      READY          1          1
  3     WAITING    RUN:cpu      READY      READY          1          1
  4     WAITING    RUN:cpu      READY      READY          1          1
  5     WAITING    RUN:cpu      READY      READY          1          1
  6*      READY    RUN:cpu      READY      READY          1
  7       READY       DONE    RUN:cpu      READY          1
  8       READY       DONE    RUN:cpu      READY          1
  9       READY       DONE    RUN:cpu      READY          1
 10       READY       DONE    RUN:cpu      READY          1
 11       READY       DONE    RUN:cpu      READY          1
 12       READY       DONE       DONE    RUN:cpu          1
 13       READY       DONE       DONE    RUN:cpu          1
 14       READY       DONE       DONE    RUN:cpu          1
 15       READY       DONE       DONE    RUN:cpu          1
 16       READY       DONE       DONE    RUN:cpu          1
 17      RUN:io       DONE       DONE       DONE          1
 18     WAITING       DONE       DONE       DONE                     1
 19     WAITING       DONE       DONE       DONE                     1
 20     WAITING       DONE       DONE       DONE                     1
 21     WAITING       DONE       DONE       DONE                     1
 22*     RUN:io       DONE       DONE       DONE          1
 23     WAITING       DONE       DONE       DONE                     1
 24     WAITING       DONE       DONE       DONE                     1
 25     WAITING       DONE       DONE       DONE                     1
 26     WAITING       DONE       DONE       DONE                     1
 27*       DONE       DONE       DONE       DONE

Stats: Total Time 27
Stats: CPU Busy 18 (66.67%)
Stats: IO Busy  12 (44.44%)
```

From this we can tell that, when process 0 first started waiting for IO, process 1 started, after process 1 ended, process 2 started, after process 2 ended, process 3 started, and after process 3, the rest of process 0 continued. And it is not a very efficient way to use system resources, since between the end of process 1 and the start of process 2, we can run process 0 once, and let it wait for IO, while process 2 runs. And do the same thing again between process 2 and process 3, this way it can save us at least 4 running time.

Q7: Here is what I got:

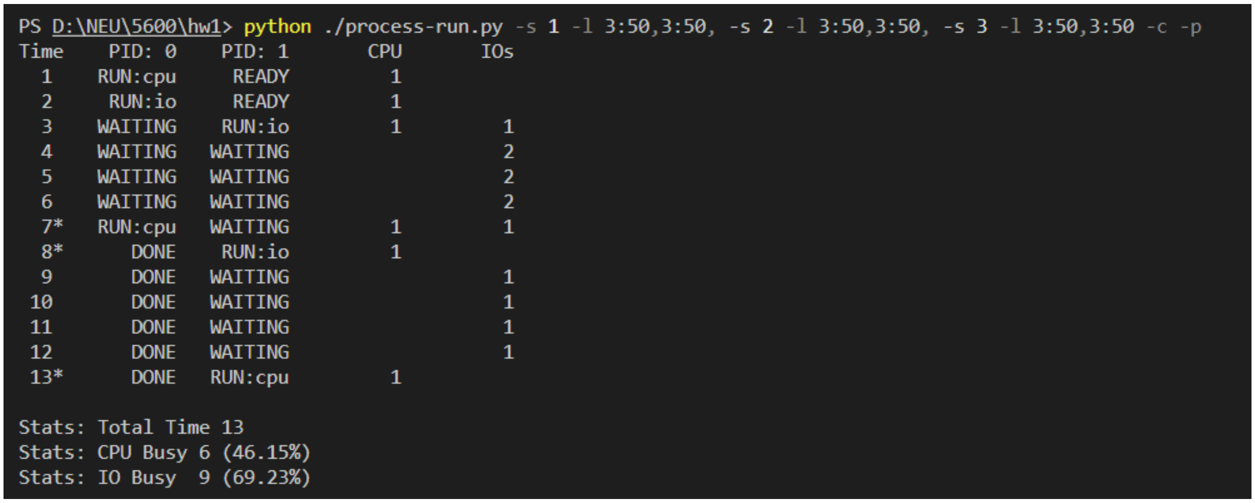


```
PS D:\NEU\5600\hw1> python ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_IMMEDIATE -c -p
Time     PID: 0     PID: 1     PID: 2     PID: 3        CPU        IOs
  1      RUN:io      READY      READY      READY          1
  2     WAITING    RUN:cpu      READY      READY          1          1
  3     WAITING    RUN:cpu      READY      READY          1          1
  4     WAITING    RUN:cpu      READY      READY          1          1
  5     WAITING    RUN:cpu      READY      READY          1          1
  6*     RUN:io      READY      READY      READY          1
  7     WAITING    RUN:cpu      READY      READY          1          1
  8     WAITING       DONE    RUN:cpu      READY          1          1
  9     WAITING       DONE    RUN:cpu      READY          1          1
 10     WAITING       DONE    RUN:cpu      READY          1          1
 11*     RUN:io       DONE      READY      READY          1
 12     WAITING       DONE    RUN:cpu      READY          1          1
 13     WAITING       DONE    RUN:cpu      READY          1          1
 14     WAITING       DONE       DONE    RUN:cpu          1          1
 15     WAITING       DONE       DONE    RUN:cpu          1          1
 16*       DONE       DONE       DONE    RUN:cpu          1
 17        DONE       DONE       DONE    RUN:cpu          1
 18        DONE       DONE       DONE    RUN:cpu          1

Stats: Total Time 18
Stats: CPU Busy 18 (100.00%)
Stats: IO Busy  12 (66.67%)
```

The differences between this time and last time are: process 0 got the priority to run first. After process 0 finished one execution, all the other processes needed to wait until process 0 started the next one. It might be a good idea simply because, comparing to last time, this one saved a lot of time, and the cpu was constantly being used. No resource was wasted.

Q8:

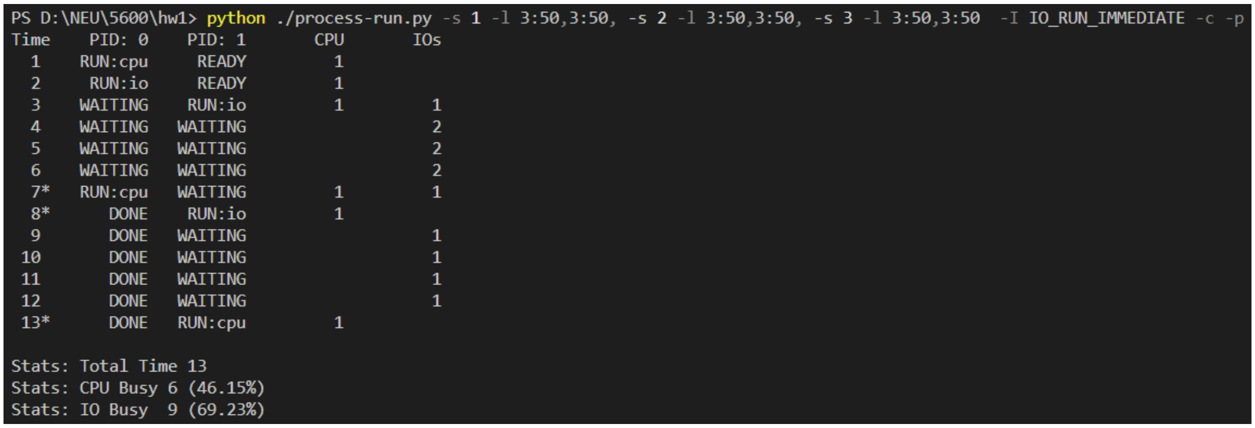
Here is what I got from random generation:



```
PS D:\NEU\5600\hw1> python ./process-run.py -s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2      RUN:io      READY          1
  3     WAITING     RUN:io          1          1
  4     WAITING    WAITING                     2
  5     WAITING    WAITING                     2
  6     WAITING    WAITING                     2
  7*    RUN:cpu    WAITING          1          1
  8*       DONE     RUN:io          1
  9        DONE    WAITING                     1
 10        DONE    WAITING                     1
 11        DONE    WAITING                     1
 12        DONE    WAITING                     1
 13*       DONE    RUN:cpu          1

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  9 (69.23%)
```

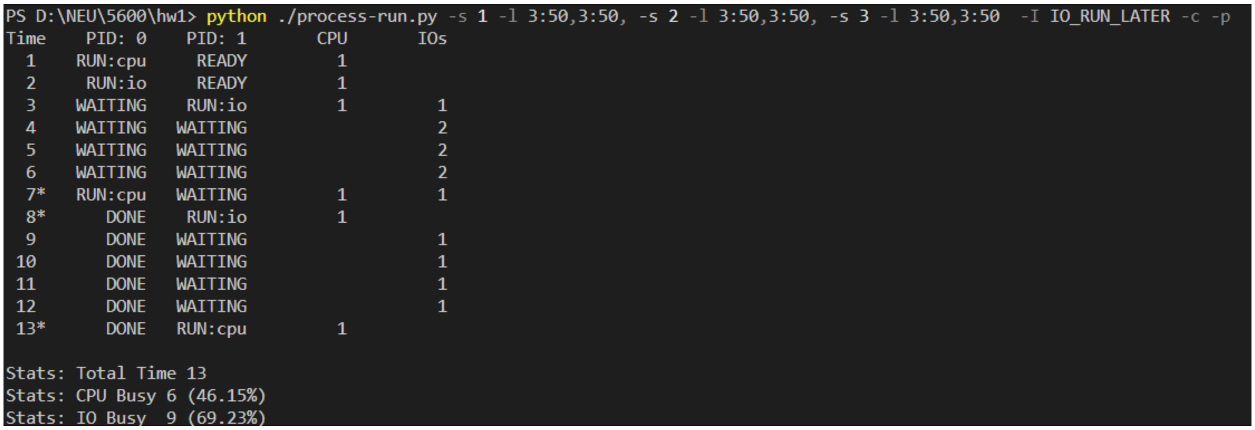
From this we can tell that: process 0 is: cpu, io, cpu, and process 1 is: io, io, cpu.

If I use “-I IO_RUN_IMMEDIATE”, nothing changes.



```
PS D:\NEU\5600\hw1> python ./process-run.py -s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50  -I IO_RUN_IMMEDIATE -c -p
Time     PID: 0     PID: 1       CPU       IOs
  1     RUN:cpu      READY         1
  2      RUN:io      READY         1
  3     WAITING     RUN:io         1         1
  4     WAITING    WAITING                   2
  5     WAITING    WAITING                   2
  6     WAITING    WAITING                   2
  7*    RUN:cpu    WAITING         1         1
  8*       DONE     RUN:io         1
  9        DONE    WAITING                   1
 10        DONE    WAITING                   1
 11        DONE    WAITING                   1
 12        DONE    WAITING                   1
 13*       DONE    RUN:cpu         1

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  9 (69.23%)
```

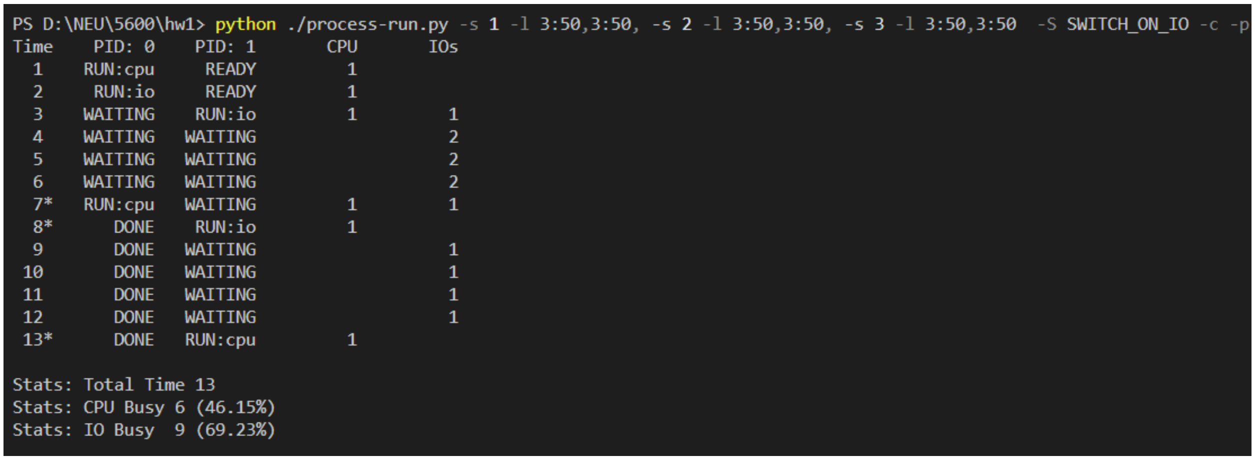
If I use “-I IO RUN LATER”,



```
PS D:\NEU\5600\hw1> python ./process-run.py -s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50  -I IO_RUN_LATER -c -p
Time     PID: 0     PID: 1       CPU       IOs
  1     RUN:cpu      READY         1
  2      RUN:io      READY         1
  3     WAITING     RUN:io         1         1
  4     WAITING    WAITING                   2
  5     WAITING    WAITING                   2
  6     WAITING    WAITING                   2
  7*    RUN:cpu    WAITING         1         1
  8*       DONE     RUN:io         1
  9        DONE    WAITING                   1
 10        DONE    WAITING                   1
 11        DONE    WAITING                   1
 12        DONE    WAITING                   1
 13*       DONE    RUN:cpu         1

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  9 (69.23%)
```

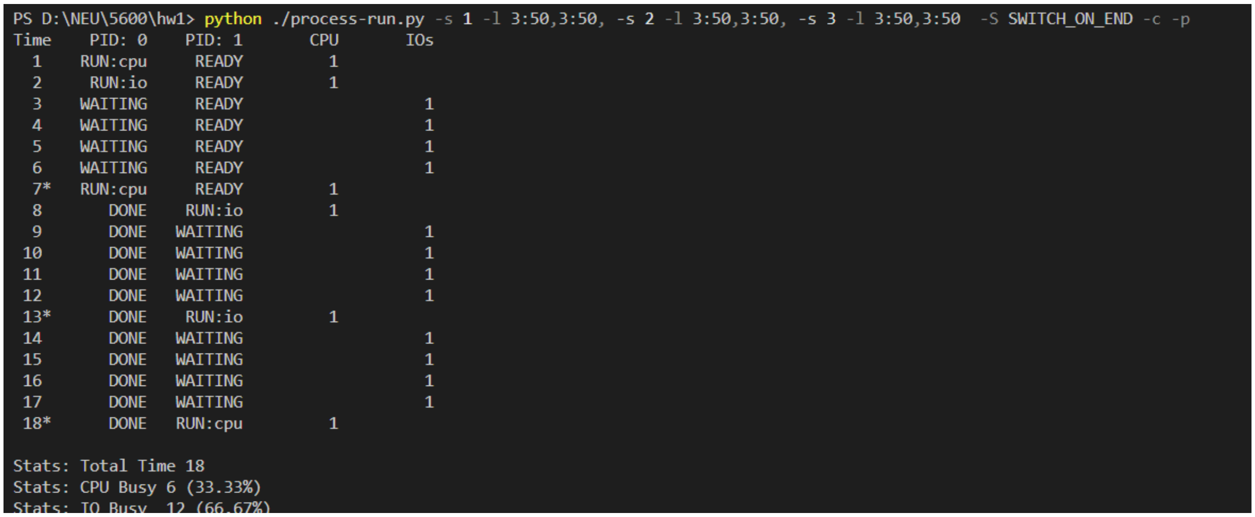
Nothing changes as well.

If I use “-S SWITCH_ON_IO”, nothing changes.



```
PS D:\NEU\5600\hw1> python ./process-run.py -s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50  -S SWITCH_ON_IO -c -p
Time     PID: 0     PID: 1       CPU       IOs
  1     RUN:cpu      READY         1
  2      RUN:io      READY         1
  3     WAITING     RUN:io         1         1
  4     WAITING    WAITING                   2
  5     WAITING    WAITING                   2
  6     WAITING    WAITING                   2
  7*    RUN:cpu    WAITING         1         1
  8*       DONE     RUN:io         1
  9        DONE    WAITING                   1
 10        DONE    WAITING                   1
 11        DONE    WAITING                   1
 12        DONE    WAITING                   1
 13*       DONE    RUN:cpu         1

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  9 (69.23%)
```

If I use ” -S SWITCH_ON_END”, when process 0 issued its first IO, process 1 had to wait until it ended.



```
PS D:\NEU\5600\hw1> python ./process-run.py -s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50  -S SWITCH_ON_END -c -p
Time     PID: 0    PID: 1       CPU       IOs
  1     RUN:cpu     READY         1
  2      RUN:io     READY         1
  3     WAITING     READY                   1
  4     WAITING     READY                   1
  5     WAITING     READY                   1
  6     WAITING     READY                   1
  7*    RUN:cpu     READY         1
  8        DONE    RUN:io         1
  9        DONE   WAITING                   1
 10        DONE   WAITING                   1
 11        DONE   WAITING                   1
 12        DONE   WAITING                   1
 13*       DONE    RUN:io         1
 14        DONE   WAITING                   1
 15        DONE   WAITING                   1
 16        DONE   WAITING                   1
 17        DONE   WAITING                   1
 18*       DONE   RUN:cpu         1

Stats: Total Time 18
Stats: CPU Busy 6 (33.33%)
Stats: IO Busy  12 (66.67%)
```