(1) 100% CPU utilization since no time CPU is idle. None of process instruction use I/O.

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

(2) ./process-run.py -l 4:100,1:0

$P_1$ ⇒ 4 instructions all CPU

$P_2$ ⇒ 1 instruction I/O

$P_1$ runs ⇒ 4

$P_2$ starts ⇒ 1 + 5 (I/O) + 1

5 ↗ default (I/O length)

Process blocked

⇒ 7

Total = 11

| 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 | BLOCKED | | 1 |
| 7 | DONE | BLOCKED | | 1 |
| 8 | DONE | BLOCKED | | 1 |
| 9 | DONE | BLOCKED | | 1 |
| 10 | DONE | BLOCKED | | 1 |
| 11* | DONE | RUN:io_done | 1 | |

```
Stats: Total Time 11
Stats: CPU Busy 6 (54.55%)
Stats: IO Busy  5 (45.45%)
```

(3) Same processes as above, order changed

↳ -L 1:0, 4:100

Should this affect things?

Yes it would because as the note "Important behaviors" states "System will switch when the current process is FINISHED or ISSUES AN IO"

Now $P_1$ is the process with I/O so when it starts and utilizes I/O the CPU is free to be used by $P_2$, using CPU for 4 instr, at that point $P_1$ one more I/O remaining

Run in using CPU, I/O, so 4 + 1 (additional

$$\text{Total} = 1 + 4 + 5 + 1$$

$$= 1 + 4 + 1 + 1$$

$$= 7$$

| Time | PID: 0 | PID: 1 | CPU | IOs |
|---|---|---|---|---|
| 1 | RUN:io | READY | 1 | |
| 2 | BLOCKED | RUN:cpu | 1 | 1 |
| 3 | BLOCKED | RUN:cpu | 1 | 1 |
| 4 | BLOCKED | RUN:cpu | 1 | 1 |
| 5 | BLOCKED | RUN:cpu | 1 | 1 |
| 6 | BLOCKED | DONE | | 1 |
| 7* | RUN:io_done | DONE | 1 | |

```
Stats: Total Time 7
Stats: CPU Busy 6 (85.71%)
Stats: IO Busy  5 (71.43%)
```

④ With "SWITCH_ON_END" the
↳ -l 1:0, 4:100 -S SWITCH_ON_END
is same as ② since no switching except the I/O process is $P_1$ in this one,

Total = 11 (same as ②)

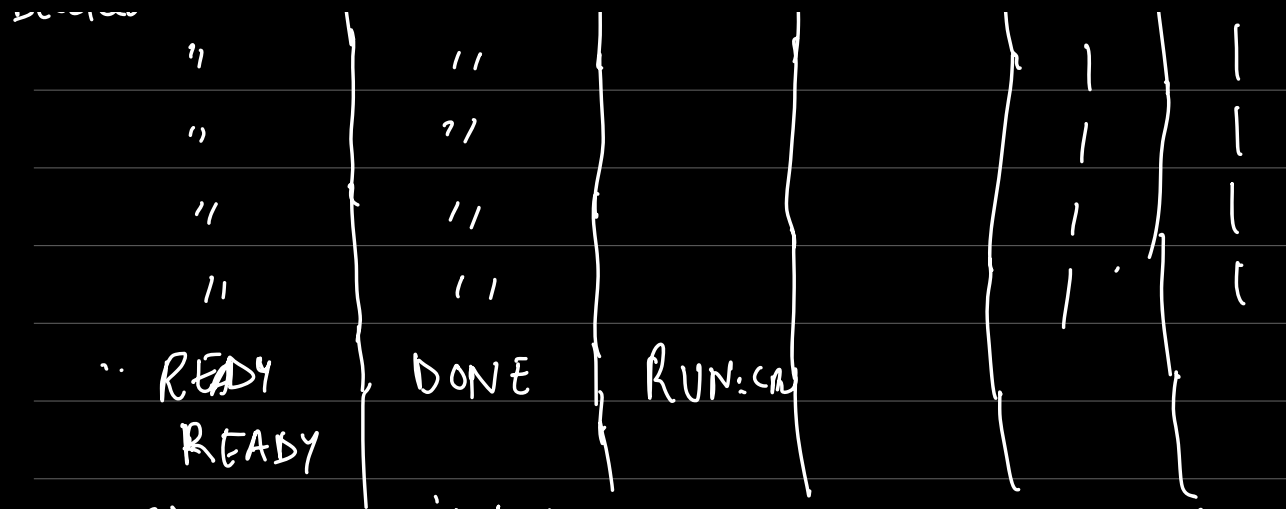
⑤ With "SWITCH_ON_IO" answer is same as ③. Default for "switch" is "SWITCH_ON_IO" hence ③ even without the "-S" flag passed explicitly.

⑥ ↳ -l 3:0, 5:100, 5:100, 5:100 -S SWITCH_ON_IO
-I IO_RUN_LATER

↓ Redundant since default

$P_0$ => 3:0 - all I/O
$P_1, P_2, P_3$ => 5:100 - all CPU

| $P_0$ | $P_1$ | $P_2$ | $P_3$ | CPU | I/O |
|---|---|---|---|---|---|
| RUN:IO | READY | READY | READY | 1 | |
| ← Blocked B | RUN:cpu | | | 1 | 1 |




Since switch is IO_RUN_LATER I'm not sure but I think $P_2$ is picked before $P_0$ even though I/O finished, leading to $P_3$ also run similarly before $P_0$. Finally $P_0$ resumes and runs remaining 2 instructions I/O, in which CPU sits idle.

Looking at the code, "next_proc" will pick $P_2$ instead of $P_0$, since that runs processes ahead of current pid first then if no processes 0 to current pid loop

Total = 1 + 5 + 5 + 5 + 1 + 7 + 7

= 31

(2 ⇒ Initiate/done I/O
5 ⇒ I/O)

```
Stats: Total Time 31
Stats: CPU Busy 21 (67.74%)
Stats: IO Busy  15 (48.39%)
```

(7) With "IO_RUN_IMMEDIATE" now in the above $P_0$ resumes after I/O completion instead $P_2$. This helps improve resource utilization since $P_0$ again starts an IO (5 steps) but CPU doesn't sit idle $P_2$ runs on it.

Why might running a process that just completed an I/O again be a good idea?

Since that process might have more I/O instructions and running those in end keeps the CPU idle which was the case in (6) with "IO_RUN_LATER"

| Time | PID: 0 | PID: 1 | PID: 2 | PID: 3 | CPU | IOs |
|---|---|---|---|---|---|---|
| 1 | RUN:io | READY | READY | READY | 1 | |
| 2 | BLOCKED | RUN:cpu | READY | READY | 1 | 1 |
| 3 | BLOCKED | RUN:cpu | READY | READY | 1 | 1 |
| 4 | BLOCKED | RUN:cpu | READY | READY | 1 | 1 |
| 5 | BLOCKED | RUN:cpu | READY | READY | 1 | 1 |
| 6 | BLOCKED | RUN:cpu | READY | READY | 1 | 1 |
| 7* | RUN:io_done | DONE | READY | READY | 1 | |
| 8 | RUN:io | DONE | READY | READY | 1 | |
| 9 | BLOCKED | DONE | RUN:cpu | READY | 1 | 1 |
| 10 | BLOCKED | DONE | RUN:cpu | READY | 1 | 1 |
| 11 | BLOCKED | DONE | RUN:cpu | READY | 1 | 1 |
| 12 | BLOCKED | DONE | RUN:cpu | READY | 1 | 1 |
| 13 | BLOCKED | DONE | RUN:cpu | READY | 1 | 1 |
| 14* | RUN:io_done | DONE | DONE | READY | 1 | |
| 15 | RUN:io | DONE | DONE | READY | 1 | |
| 16 | BLOCKED | DONE | DONE | RUN:cpu | 1 | 1 |
| 17 | BLOCKED | DONE | DONE | RUN:cpu | 1 | 1 |
| 18 | BLOCKED | DONE | DONE | RUN:cpu | 1 | 1 |
| 19 | BLOCKED | DONE | DONE | RUN:cpu | 1 | 1 |
| 20 | BLOCKED | DONE | DONE | RUN:cpu | 1 | 1 |
| 21* | RUN:io_done | DONE | DONE | DONE | 1 | |

Stats: Total Time 21
Stats: CPU Busy 21 (100.00%)
Stats: IO Busy 15 (71.43%)

(8) -s 1 -l 3:50, 3:50

seed

$P_0$ $P_1$ => 3:50 could mean few combination

- all 3 CPU
- all 3 I/O
- 2 CPU, 1 I/O $\Rightarrow P_1$
- 1 I/O, 1 CPU $\Rightarrow P_0$

| $P_0$ | $P_1$ | CPU | I/O |
|---|---|---|---|
| RUN: io | READY | 1 | |
| BLOCKED | RUN: cpu | 1 | 1 |
| " | RUN: cpu | 1 | 1 |
| " | RUN: io | 1 | 2 |
| " | BLOCKED | | 2 |
| " | " | | 2 |
| RUN: io_done | " | 1 | 1 |
| BLOCKED | " | | 2 |
| " | " | | 2 |
| " | | | 1 |
| " | | | 1 |
| " | | | 1 |
| RUN: io_done | | 1 | |
| RUN: cpu | | 1 | |

With seed 3 -l 3:50, 3:50

| Time | PID: 0 | PID: 1 | CPU | IOs |
|---|---|---|---|---|
| 1 | RUN:cpu | READY | 1 | |
| 2 | RUN:io | READY | 1 | |
| 3 | BLOCKED | RUN:io | 1 | 1 |
| 4 | BLOCKED | BLOCKED | | 2 |
| 5 | BLOCKED | BLOCKED | | 2 |
| 6 | BLOCKED | BLOCKED | | 2 |
| 7 | BLOCKED | BLOCKED | | 2 |
| 8* | RUN:io_done | BLOCKED | 1 | 1 |
| 9* | RUN:cpu | READY | 1 | |
| 10 | DONE | RUN:io_done | 1 | |
| 11 | DONE | RUN:io | 1 | |
| 12 | DONE | BLOCKED | | 1 |
| 13 | DONE | BLOCKED | | 1 |
| 14 | DONE | BLOCKED | | 1 |
| 15 | DONE | BLOCKED | | 1 |
| 16 | DONE | BLOCKED | | 1 |
| 17* | DONE | RUN:io_done | 1 | |
| 18 | DONE | RUN:cpu | 1 | |

Stats: Total Time 18
Stats: CPU Busy 9 (50.00%)
Stats: IO Busy 11 (61.11%)

With "IO_RUN_IMMEDIATE", the CPU utitization is slightly better, for this config of processes (1 time less)

| Time | PID: 0 | PID: 1 | CPU | IOs |
|---|---|---|---|---|
| 1 | RUN:cpu | READY | 1 | |
| 2 | RUN:io | READY | 1 | |
| 3 | BLOCKED | RUN:io | 1 | 1 |
| 4 | BLOCKED | BLOCKED | | 2 |
| 5 | BLOCKED | BLOCKED | | 2 |
| 6 | BLOCKED | BLOCKED | | 2 |
| 7 | BLOCKED | BLOCKED | | 2 |
| 8* | RUN:io_done | BLOCKED | 1 | 1 |
| 9* | READY | RUN:io_done | 1 | |
| 10 | READY | RUN:io | 1 | |
| 11 | RUN:cpu | BLOCKED | 1 | 1 |
| 12 | DONE | BLOCKED | | 1 |
| 13 | DONE | BLOCKED | | 1 |
| 14 | DONE | BLOCKED | | 1 |
| 15 | DONE | BLOCKED | | 1 |
| 16* | DONE | RUN:io_done | 1 | |
| 17 | DONE | RUN:cpu | 1 | |

Stats: Total Time 17
Stats: CPU Busy 9 (52.94%)
Stats: IO Busy 11 (64.71%)

With "SWITCH_ON_END", utilization is reduced than before because $P_0$ doesn't switch to $P_1$, on start I/O instruction

# making P1's I/O instruction not to start

| Time | PID: 0 | PID: 1 | CPU | IOs |
|---|---|---|---|---|
| 1 | RUN:cpu | READY | 1 | |
| 2 | RUN:io | READY | 1 | |
| 3 | BLOCKED | READY | | 1 |
| 4 | BLOCKED | READY | | 1 |
| 5 | BLOCKED | READY | | 1 |
| 6 | BLOCKED | READY | | 1 |
| 7 | BLOCKED | READY | | 1 |
| 8* | RUN:io_done | READY | 1 | |
| 9 | RUN:cpu | READY | 1 | |
| 10 | DONE | RUN:io | 1 | |
| 11 | DONE | BLOCKED | | 1 |
| 12 | DONE | BLOCKED | | 1 |
| 13 | DONE | BLOCKED | | 1 |
| 14 | DONE | BLOCKED | | 1 |
| 15 | DONE | BLOCKED | | 1 |
| 16* | DONE | RUN:io_done | 1 | |
| 17 | DONE | RUN:io | 1 | |
| 18 | DONE | BLOCKED | | 1 |
| 19 | DONE | BLOCKED | | 1 |
| 20 | DONE | BLOCKED | | 1 |
| 21 | DONE | BLOCKED | | 1 |
| 22 | DONE | BLOCKED | | 1 |
| 23* | DONE | RUN:io_done | 1 | |
| 24 | DONE | RUN:cpu | 1 | |

Stats: Total Time 24
Stats: CPU Busy 9 (37.50%)
Stats: IO Busy  15 (62.50%)