Homework (week 1)

▼ Q1

▼ What should the CPU utilization be?

The cpu time consumption would be 100%, because the flag we put in: "5:100,5:100" refers that this process is totally CPU-consuming but not have IO related task.

▼ Why do you know this?

Because from the explanation of flag '-l', we know Y stands for the percentage of CPU-related task.

▼ Q2

▼ How long does it take to complete both processes?

9 clock tick would be required. Because after 4 clock tick of CPU task of the first process, the system would switch to next process and execute its first instruction, which starts an IO task and 5 clock tick of IO waiting would be required as mentioned in flag "**iolength**".

> 💡 **I think the provided answer 10 clock tick is incorrect, because the last tick is not occupied at all.**

▼ Q3

▼ What happens now?

The same amount of task as in Q2 are finished in almost half time: **5 clock tick**.

▼ Does switching the order matter?

Yes, in this case, switching the order helps.

▼ Why?

It is because the scheduling algorithms has specified that by default "**SWITCH_ON_IO**". It means that when encounters an IO task, the system would switch to another process in order to avoid a long IO waiting time.

▼ Q4

▼ With the flag set to SWITCH_ON_END, what happens when you run the following two processes one doing I/O and the other doing CPU work?

In this case, the system would have to wait until the IO task has finished then start execute instruction of next process. Therefore the time required would be **9 clock tick** again.

▼ Q5

▼ What happens now?

It only takes 5 clock ticks to finish the task.

▼ Q6

▼ What happens when you run this combination of processes?

Once the first time the IO finished, it is not executed immediately but executed on every other processes are finished.

▼ Are system resources being effectively utilized?

Definitely not. The IO resources are not occupied when there are still lots of IO tasks.

▼ Q7

▼ How does this behavior differ?

Once the IO task is finished, next instruction with IO-related task would be immediately executed rather than wait until every other process finished.

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

There might be several reasons:

- As the question mentioned, for those IO intensive task, it is better to keep IO resources fully occupied/used through frequently check whether for the same process have following IO tasks.

- The second reason would be from the aspect of user experience. Lets say a person is browsing a website and have waited for 1 second for the page loading and page rendering. Then he would probably want to start doing things immediately rather than waiting for more time (though the CPU nowadays are too fast to let the user feel the delay led by switching to and execute another process)

▼ Q8

▼ What happens when you use the flag -I IO_RUN_IMMEDIATE vs. -I IO_RUN_LATER?

In this case, the change on the strategy does not change the overall time a lot, e.g. most of situation they have spent the same amount of time.

> 💡 But I had tried on if one of the process is heavy on CPU task, then the difference is huge. Perhaps IO_RUN_IMMEDIATE is more suitable with task with more CPU workload, because in this case, the strength of high resource utilization rate play a role.

▼ What happens when you use -S SWITCH_ON_IO vs. -S SWITCH_ON_END

In common sense, SWITCH_ON_IO is faster than SWITCH_ON_END, because it does not wait for the IO task idly, but switch to another task. And the experiment proves my expectation, in most cases SWITCH_ON_IO finishes task within a shorter time.