Exercise sheet 11 – Scheduling

Goals:

- Scheduling strategy calculations
- Scheduling strategy knowledge
- Scheduling on Linux (optional)

Exercise 11.1: Scheduling strategy calculations

There are five background jobs/processes (A - E), which are at the same time given to a computer for processing. Arrival order, processing time, and priority for each job is stated in the table (5 is the highest priority).

Arrival order	Job	Processing time (in min)	Priority
1	A	10	3
2	B	6	5
3	C	2	2
4	D	4	1
5	E	8	4

Calculate the mean residence time for each job for each of the following algorithms! You don't need to consider the time needed for changing the process and I/O (context switch). In exercises b) to e) it is expected, that the job is completed, before it is changed to the next one (non-preemptive). Consider a single core CPU for this exercise.

(a) Round Robin with processor sharing. Use 2 min. as time slice.

Proposal for solution:

Schedule order	A	B	**C**	D	E	A	B	**D**	E	A	**B**	E	A	**E**	**A**
Time steps	2	4	**6**	8	10	12	14	**16**	18	20	**22**	24	26	**28**	**30**
Time left	8	4	0	2	6	6	2	0	4	4	0	2	2	0	0
End			↑					↑			↑			↑	↑

Mean residence time: $(30 + 22 + 6 + 16 + 28)/5 = 20.4$

(b) Priority based scheduling

Proposal for solution:

Job (processing time):	B (6)	E (8)	A (10)	C (2)	D (4)
Job end time:	6	14	24	26	30

Mean residence time: $(6 + 14 + 24 + 26 + 30)/5 = 20$

(c) First come first served (FCFS)

Proposal for solution:

Job (processing time):	A (10)	B (6)	C (2)	D (4)	E (8)
Job end time:	10	16	18	22	30

Mean residence time: $(10 + 16 + 18 + 22 + 30)/5 = 19,2$

(d) Shortest job first (SJF)

Proposal for solution:

Job (processing time):	C (2)	D (4)	B (6)	E (8)	A (10)
Job end time:	2	6	12	20	30

Mean residence time: $(2 + 6 + 12 + 20 + 30)/5 = 14$

(e) Shortest job first (SJF) with new arrival time: (A: 0, B: 2, C: 3, D: 11, E:14)

Proposal for solution:

Job	Arrival	Processing time	Start	End	Response time	Schedule order
A	0	10	0	10	10	1.
B	2	6	16	22	20	4.
C	3	2	10	12	9	2.
D	11	4	12	16	5	3.
E	14	8	22	30	16	5.

Mean residence time: $(10 + 20 + 9 + 5 + 16)/5 = 12$

Exercise 11.2: Scheduling strategy knowledge

Hint: Please give a suitable reason for your answers.

(a) Is the FCFS scheduling strategy well suited for desktop systems?

Proposal for solution: No, because long running tasks will block short tasks (e.g. mouse movements) and users would think that the PC isn't responding.

(b) For what is SJF optimised?

Proposal for solution: SJF is optimised for throughput: Process as much tasks as possible. But it can happen that long running processes will not be processed (starvation).

(c) Why is RR better suited for desktop systems than FCFS?

Proposal for solution: Because a processes is served for a limited time one after another: This allows smaller tasks to also be served (e.g. mouse movements) and therefore a user has faster response to his actions.

(d) Why is EDF better suited for real-time systems than FCFS?

Proposal for solution: Because it takes into account when a task has to be finished. FCFS does start one after another (if one has finished), but this can be too late for some tasks.

(e) Which scheduling strategy is usually used in shops when you want to pay your goods? Is this the best strategy to serve as much customers as possible?

Proposal for solution: FCFS. No, because it does not take into account how much goods someone have and therefore how long it takes to do the payment.

Exercise 11.3: Scheduling on Linux (optional)

(a) Read about the Linux CFQ (completely fair queuing) scheduler. Here are some literature points:

- `https://www.thomas-krenn.com/de/wiki/Linux_I/O_Scheduler#CFQ`

- `https://en.wikipedia.org/wiki/Completely_Fair_Scheduler`
- `https://www.phoronix.com/scan.php?page=article&item=linux_2637_video&num=1`
- `http://www.ece.ubc.ca/~sasha/papers/eurosys16-final29.pdf`
- `https://documentation.suse.com/sles/12-SP4/html/SLES-all/cha-tuning-taskscheduler.html`

(b) How does it work? Which criteria are taken into account for scheduling?

(c) How does it work on single core CPUs?

(d) How does it work on multi core core CPUs?

(e) Is the CFQ scheduler suitable for real-time systems?