Laboratory work No. 4

CPU scheduling

1. **Aim of the work**: Investigate CPU job scheduling algorithms

2. The Theoretical Part

2. 1. First Come, First Served Scheduling Algorithm

Simplest CPU scheduling algorithm is the *first-come, first-served* (FCFS) scheduling algorithm. With this scheme, the process that requests the CPU first is allocated to the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. The running process is then removed from the queue.

FCFS algorithms average waiting time:

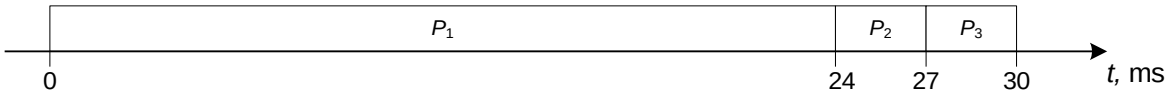
$$\overline{t_l} = \sum_{i=1}^{n} t_{l_i}\,, \tag{1}$$

here t_{l_i} – i-th process waiting time; n – number of processes.

The average waiting time under the FCFS policy is often quite long. Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time, ms
P_1	24
P_2	3
P_3	3

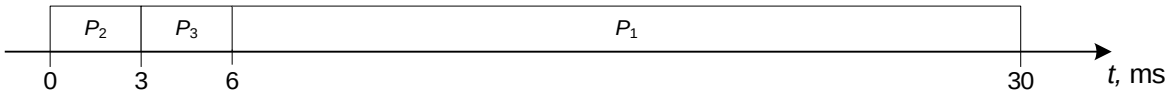
If the processes arrive in the order – P_1, P_2, P_3 and are served in FCFS order, we get the result shown in the following Gantt chart:



The waiting time for P_1 process is 0 ms; 24 milliseconds for P_2 process and 27 milliseconds for P_3 process. Thus, the average waiting time:

$$\frac{0+24+27}{3} = 17\text{ms}\,.$$

But if the processes arrive in the order P_2, P_3, P_1 the results will be as shown in the following Gantt chart:



In this case, the average waiting time is:

$$\frac{0+3+6}{3} = 3\text{ ms}\,.$$

This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the process's CPU burst times vary greatly.

2. 2. Shortest Job First Scheduling Algorithm

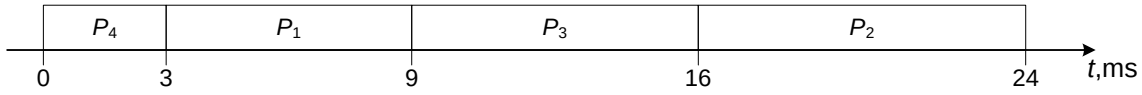
A different approach to the CPU scheduling is the *shortest job first* (SJF) scheduling algorithm. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

The SJF scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes. The real difficulty with the SJF algorithm is knowing the length of the next CPU request.

Consider the following set of processes:

Process	Burst Time, ms
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF scheduling, these processes will be scheduled according to the following Gantt chart:



The waiting time for P_1 process is 3 milliseconds; for P_2 process – 16 ms; for P_3 process – 9 ms; for P_4 process – 0 ms. Thus, the average waiting time is:

$$\frac{3+16+9+0}{4} = 7 \text{ ms}.$$

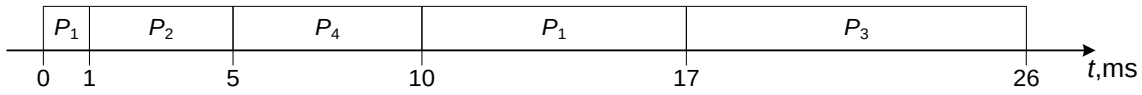
2. 3. Shortest Remaining Time First Scheduling Algorithm

The SJF algorithm can be either pre-emptive or non pre-emptive. The choice arises when a new process arrives at the ready queue while a previous process is still executing. The next CPU burst of the newly arrived process may be shorter then what is left of the currently executing process. A pre-emptive SJF algorithm will pre-empt the currently executing process, whereas a non pre-emptive SJF algorithm will allow the currently running process to finish its CPU burst. Pre-emptive SJF scheduling is sometimes called shortest *remaining time first* (SRTF) scheduling algorithm.

As an example, consider the following four processes, with the length of the CPU burst given in milliseconds:

Process	Arrival Time, ms	Burst Time, ms
P_1	0	6
P_2	1	8
P_3	2	7
P_4	3	3

If the processes arrive at the ready queue at the times shown and need the indicated burst times, then the resulting pre-emptive SJF schedule is as depicted in the following Gantt chart:



Process P_1 is started at time 0, since it the only process in the queue. Process P_2 arrives at time 1 ms. The remaining time for process P_1 is 7 ms, which is larger when the time required by process P_2 – 4 milliseconds, so process P_2 is scheduled. The waiting time for P_1 process is $(10-1)$ milliseconds; for P_2 process – $(1-1)$ ms; for P_3 process – $(17-2)$ ms; for P_4 process – $(5-3)$ ms. The average waiting time for this example:

$$\frac{(10-1)+(1-1)+(17-2)+(5-3)}{4} = 6,5 \text{ ms}.$$

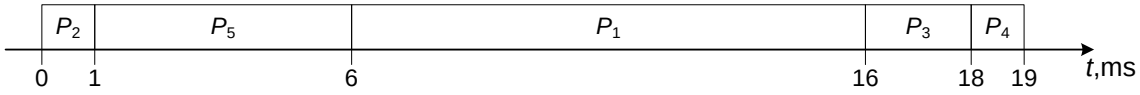
2. 4. Priority Scheduling Algorithm

In *Priority scheduling* (PS) algorithm a priority is associated with each process, and the CPU is allocated to the process with the highest priority. In our case lower number is higher priority. Equal-priority processes are scheduled in FCFS order.

As an example, consider the following five processes with the length of the CPU burst given in milliseconds:

Process	Burst Time, ms	Priority
P_1	10	3
P_2	1	1
P_3	2	5
P_4	1	4
P_5	5	2

Using priority scheduling, we would schedule these processes according to the following Gant chart:



Process P_2 begins at time 0 ms, because its priority is the highest. Next CPU executes P_5 process, because its priority is 2. The average waiting time of this algorithm is:

$$\frac{6+0+16+18+1}{5} = 8,2 \text{ ms}.$$

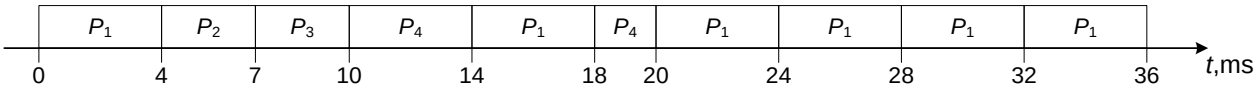
2. 5. Round Robin Scheduling

The *round robin* (RR) algorithm is similar to FCFS scheduling algorithm, but pre-emption is added to switch between processes. A small unit of time, called a time quantum, is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

As an example, consider that a time quantum is 4 ms ant the following five processes with the length of the CPU burst given in milliseconds:

Process	Burst Time, ms
P_1	24
P_2	3
P_3	3
P_4	6

Using RR scheduling, these processes will be scheduled according to the following Gantt chart:



The average waiting time:

$$\frac{(20-8)+4+7+(18-4)}{4} = 9,25 \text{ ms}.$$

Turnover time is time from process appearance on ready queue till the end of process execution. This algorithms average turnover time is:

$$\frac{36+7+10+20}{4} = 18,25 \text{ ms}.$$

3. Work task

Write code which illustrates CPU scheduling algorithms described in 2.1 – 2.5. In laboratory work will be presented: process number, CPU burst times, arrival times (in SRTF algorithm), priorities (in PS algorithm) and time quantum (in RR algorithm).

It may be assumed that all processes occur at zero time (except SRTF algorithm), but in certain order, such as: P_1, P_2 ... P_n. It must be possible to change number of processes and their CPU burst time.

Must be calculated and displayed: each process waiting time, average waiting time, each process turnover time, average turnover time.

Literature

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. Applied Operating Systems Concepts. 2000, 840 p. ISBN 0-471-36508-4.