Operating Systems Lab 2: Scheduling and Page Replacement

The second lab for Operating Systems will have two assignments: one about scheduling and one about page replacement. The page replacement assignment asks you to implement two algorithms, each worth 25% of your lab grade, while the scheduling assignment counts for the remaining 50% of the grade.

Scheduling

In the first exercise you are going to make your own virtual scheduler. You will make a simple non-pre-emptive First-Come-First-Serve scheduler.

Remember that most processes are a mix of CPU and I/O. Assume that there is only one CPU and one I/O device, so only one process can do CPU work and one process can do I/O work at any moment. The scheduler will receive its input on `stdin`, which will look like:

```
0 100 25 100 20 50 20 100 10 200 -1
15 30 15 30 10 40 10 50 -1
...
```

Each line represents one process. The first column will be the arrival time of the process. After that, there are alternating CPU- and I/O-times, starting with CPU time. A -1 indicates the end of a process. So, the first process in this example will arrive at time `0`, then spend `100` ticks on CPU work, then `25` ticks on I/O, etc.

When all processes are done, the scheduler should print the average turnaround time for all processes to `stdout` (do not forget the newline!). This number will be a `double`, when printing make sure to truncate the number to only print an integer, for example using `printf("%.0lf\n", ...);`. Remember that turnaround time is the time between creation of the process (i.e. entering the queue) and finishing the process.

In the case that at a given time `t`, a new process is scheduled to start (i.e. the first column contains value `t`) and an existing process finishes an I/O block (i.e. it should be queued for CPU time), first the new process should be queued, and then the existing process. In other words: newly arriving processes are treated with higher priority than existing processes. The arrival times of processes are unique and increasing line-by-line.

Page Replacement Algorithms

In this second exercise you are going to implement your own page replacement algorithms. These will be the FIFO page replacement algorithm and the clock page replacement algorithm. You should make two separate implementations to hand in on Themis.

The program will receive the input on its `stdin`, for example:

```
4
1 2 3 4 7 2 5 2 3 1 7 6 4
```

The first line indicates the number of physical memory frames. After that, there will be a stream of page references. You may assume a maximum of 10 frames, 50 page numbers and 100 references (i.e. storing this in an `int` should work fine and you should not worry about overflows). When the program is finished, it should print the number of page faults to `stdout`, as just a number with a newline.