Department of "Computer Engineering"

"Operating Systems"

Project Report

- Event Simulator -

Lecturer: Prof. Dr. Ercan SOLAK

Leyla Abdullayeva - 1904010038
Darına Karimova - 2104011002

Project Description

Our program simulates some aspects of operating systems. The entire simulation is predicated on the text inputs that the program receives from the user.

The program should simulate the demand paging memory management.
When a replacement process appears within the system, it's not loaded into memory.

At the beginning , the program asks the user three questions:

- How much RAM memory is there on the simulated computer?
  The program receives the amount in bytes (not kilobytes). I can enter any number up to 4000000000 (4 billion).

- What is the size of a page/frame?
  The enumeration automatically begins with 0.

- How many hard disks should be in the simulated computer?
  The enumeration starts with 0 as well.

```
int main() {

    unsigned int RAM = 0;
    int numberHardDisks = 0;
    unsigned int pageSize = 0;

    cout << "How much RAM is there?" << endl;
    std::cin >> RAM;

    cout << "What is the size of a page/frame?" << endl;
    cin >> pageSize;

    cout << "How many hard disks does the simulated computer have?" << endl;
    cin >> numberHardDisks;
```

Then the program creates an initial process, which is a system process that cannot be terminated. All other processes in the system are the descendants of this initial one. In our simulation only, this process never uses the CPU or memory.

After that, the simulation begins. The program implements according to the user's inputs. The system does not ask for the confirmation to enter another input. The user inputs signal some system events.

The possible inputs are:

‘A’ input means a replacement process has been created. When the new process has been created no memory is allocated for it. The program places the newly created process at the end of the ready-queue.

The PID for the new process starts from 2 because the first one belongs to the first system process.

'S r' displays what process is currently using the CPU and what processes are waiting within the ready-queue.

'fork' should be a child process which is placed at the end of the ready-queue.

'toQueue' moves the process currently running in the CPU to the end of the ready-queue. In case if there is one process in the ready queue, it places a new process into the CPU.

```
    System os(RAM, pageSize, numberHardDisks);

    string input;
    cin>>input;

    //while(1) {
        //to display which process is using the CPU and which is waiting in the ready queue
        if (input == "S r") {
            os.showCurrentProcess();
        }

        //the process using the CPU forks a child.
        //the child is places in the end of the ready queue
        else if (input == "fork") {
            //os.fork();
        }

        //to move the current process in CPU to the end of the ready queue
        //plus, places a new process into the CPU
        else if  (input == "toQueue"){
            os.fromCPUtoReadyQ();
        }

        //to create a process and places at the end of ready queue
        // if the ready queue is empty it places the process in the CPU
        else if (input == "A"){
            os.createProcess();
        }
    //}

}
```