Ministry of Education, Culture and Research

Technical University of Moldova

Faculty of Computers, Computer Science and Microelectronics

**REPORT**Operating systems: internal mechanisms and design princiles

Topic: *OS Simulation*

Written by: Serghei Derevenco

Group: FAF-182

Checked by: Calin Rostislav

Chisinau 2020

**Task:**

Create an application which has the functionalities of a simple operating system. These should include the following 4 phases:

Phase 1: Checking Hardware.

Phase 2: Loading the Operating System.

Phase 3: OS started (prompting user/password).

Phase 4: Replacing some useful, already existent commands in the CLI.

*For this application I used the Rust programming language (Version - rustup 1.21.1 (7832b2ebe 2019-12-20)). The system for wrritting the code was Intellij IDE (Version - 2019 3.2) from JetBrains with compiler for Rust.*

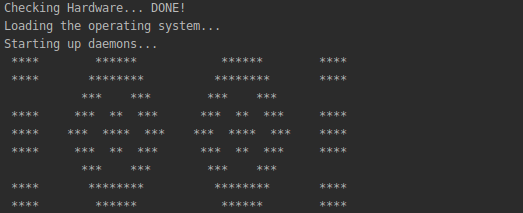
***Code of application:***

extern crate os\_type;use std::io;use std::io::{Write, stdout};use crossterm::{ ExecutableCommand,};use std::fs::{read\_dir};use std::{thread, time};fn os\_loading() { println!("Checking Hardware... DONE!"); thread::sleep(time::Duration::*from\_secs*(1)); println!("Loading the operating system..."); thread::sleep(time::Duration::*from\_secs*(1)); println!("Starting up daemons..."); thread::sleep(time::Duration::*from\_secs*(1)); println!(" \*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\* "); println!(" \*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\* "); println!(" \*\*\* \*\*\* \*\*\* \*\*\* "); println!(" \*\*\*\* \*\*\* \*\* \*\*\* \*\*\* \*\* \*\*\* \*\*\*\* "); println!(" \*\*\*\* \*\*\* \*\*\*\* \*\*\* \*\*\* \*\*\*\* \*\*\* \*\*\*\* "); println!(" \*\*\*\* \*\*\* \*\* \*\*\* \*\*\* \*\* \*\*\* \*\*\*\* "); println!(" \*\*\* \*\*\* \*\*\* \*\*\* "); println!(" \*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\* "); println!(" \*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\* "); thread::sleep(time::Duration::*from\_secs*(1));}enum Functions { *Help*, *TypeOS*, *Ls*, *Clear*, *Quit*,}fn main() { os\_loading(); let mut pass = String::*new*(); loop { println!("Please log in:"); print!("> "); io::stdout().flush().unwrap(); io::stdin().read\_line(&mut pass).expect("Something went wrong"); pass = pass.trim().to\_string(); match pass.as\_str() { "admin" => { println!("Welcome!"); break; }, \_ => { println!("Wrong, please try again!"); pass = String::*new*(); } } } loop { print!("> "); io::stdout().flush().unwrap(); let mut inp = String::*new*(); io::stdin().read\_line(&mut inp).expect("Something went wrong..."); let args: Vec<&str>; if inp.contains("do") { args = inp.trim().splitn(3, ' ').collect(); } else { args = inp.trim().splitn(2, ' ').collect(); } let command = match args[0]{ "ls" => Functions::*Ls*, "clear" => Functions::*Clear*, "quit" => Functions::*Quit*, "os-info" => Functions::*TypeOS*, "help" => Functions::*Help*, \_ => { println!("You need to provide an available command"); Functions::*Help* }, }; match command { Functions::*Help* => { println!(); println!("Available commands: "); println!(); //help println!("'help' Displays available commands."); println!(); //os-info println!("'os-info' Displays the current OS version."); println!(); //ls println!("'ls' Displays a list of files from the directory."); println!(); //clear println!("'clear' Clears the screen."); println!(); //quit println!("'quit' Shuts down the operating system."); println!(); },

Functions::*TypeOS* => { let os = os\_type::current\_platform(); println!("Type: {:?}", os.os\_type); println!("Version: {}", os.version); }, Functions::*Clear* => { print!("\x1B[2J"); stdout().execute(crossterm::cursor::MoveTo(0, 0)).expect("Something went wrong"); }, Functions::*Ls* => { let paths = read\_dir("files/").unwrap(); for path in paths { println!("{}", path.unwrap().path().display()); } }, Functions::*Quit* => { println!("Goodbye!"); thread::sleep(time::Duration::*from\_secs*(1)); break; } \_=> {} } }}

**Output:**

Phase 1-2:

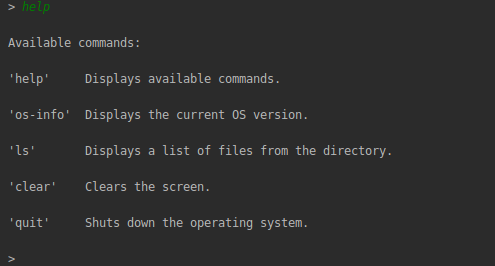


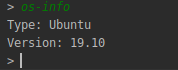
Phase 3:

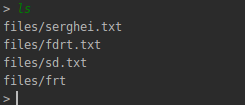


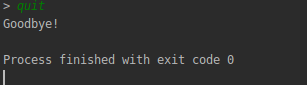
Phase 4:

I implemented 5 commands:









**Conclusion:**

In conclussion I can say that this laboratory gave a big boost to the study of writing my own operating system. For the application I used the Intellij IDE to compile my code in Rust programming language, this IDE has a good interface that allowed me to work comfortable and fast. I think that it was a great opportunity to develop my skills in OS developing and to learn a new programming language.