

**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

CSE 3033

Operating Systems

Project 2

Date: 19.12.2022

|  |  |  |
| --- | --- | --- |
|  | Student Id | Name Surname |
| 1 | 150119743 | Burak DURSUN |
| 2 | 150119739 | Emir Ege EREN |
| 3 | 150119745 | Haydar Taha TUNÇ |

**Introduction**

The purpose of this program is to write a terminal application in C. Our terminal application waits for the user to enter an input, and as a result of the analysis it makes within itself, it is expected to give the correct output to the user.

**Part A:**

In part A of the program, it asks us to create a new process. Before creating this process, the input program received from the user searches at $PATH. It uses the **getenv** method to get the folders in $PATH. Then, after separating the folders, it verifies the location of the program with the access method. Then it uses the fork method to create a new process. If the **pid** is less than 0, it runs our proceess using the **execv** method. If the user adds the ampersand ("&") character to the end of the input, the parent process does not wait for the child process it created to finish. If there is no input, it waits for the process it has created to finish using the **waitpid** method.

**Sample Screenshot **

**Part B**

In the B part of the program, the program asks us to store each process we created, when we created it, the process number and what we gave as input. We need to give the history input to show the stored commands on the screen. After entering this command, the program prints the last 10 processes in the linked list. If the user wants to view a specific process, the user must enter the "history -i num" command. If the user enters a number greater than 10, the program will send him an error message.

**Sample Screenshots**



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

We used linked list in the background processes of the program. In this linked list, there are the process ids of the background processes running.

**Sample Screenshot**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Finally, in part B, the program closes itself after the user gives the exit entry to the program.

**Sample Screenshot**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Part C**

In the C part of the program, we added the I/O Redirection feature to our program. These are writing data to the file, writing data to the end of the file, taking input from the file and writing the output of the program it runs using the input from the file to another file, etc. features. We used the **dup2** method to give these inputs and outputs. We stored the files in **int fd**. Using the **open** method, we opened the files in the mode we wanted and did the necessary operations.

**Sample Screenshots**

**Output to file.**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Output to at the end of the file.**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Input from file.**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ERROR**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Take input from file and write to output file.**

