**University of Cyprus**

ECE316 – Operating Systems and Networks Laboratory

*Assignment 6*

*Due Date: 10/03/2022*

**Name:** Theodosios Ioannou **ID: 1020844**

Team 7

Edwina Karoulla **ID: 1042364**

Foivos Lympouras **ID:1016477**

Stelios Karagiorgis **ID:1021344**

**Exercise 1:**

**socket ():** This command is used as a syscall in order to create a new socket which is one end of intracommunication. It receives as arguments the following:

* 1. socket’s address domain
  2. socket type
  3. socket’s protocol

**bind ():** This command sets address and the size of it bytes to the socket when it is created as when it does so its address is undefined.

**listen ():** This command sets the limit of the queue for the incoming connection requests. It receives as arguments the following:

* 1. socket
  2. max queue size (for pending connection requests)

**accept ():** This command exports the first request for connection from the pending connections (listen ()). It creates a new connected socket and returns a new file which refers to it. It receives as arguments the following:

* 1. socket
  2. address\_length
  3. socket\_address

**connect ():** This fuction works as a syscall which connects a socket with its given address. In order to define the addressing type, the address size in bytes must be also given to it. In this fashion, we establish the start of communication. It receives as arguments the following:

* 1. socket
  2. address\_length
  3. serv\_addr

**serve():** This command reads the client’s message and prints it. Afterwards, when “Enter” is pressed by the user, each character is read and gets sent to the server.

**read ():**  This command reads a set amount of bytes (set by the count) from a file descriptor and places them in a buffer. It receives as arguments the following:

* 1. file descriptor
  2. count
  3. buf

**write ():**This command writes count amount of bytes in the file given by the file descriptor starting from the buf value of the buffer. A read command is expected to follow afterwards (after the write () command). It receives as arguments the following:

1. file descriptor
2. count
3. buf

**close ():** This command is used to disable any mentions to a file so that it can be reused when the file descriptor is closed. It receives as arguments the following:

1. file descriptor

**Exercise 2:**

**Objective:**

1. Create a program that implements the stop and wait protocol.
2. Receive an input string from the user in the client socket.
3. Each character of the string input gets sent to the server with its sequence number (0 or 1 as char), 2 chars in total for each packet
4. Display each packet and acknowledgement sent and received in the channel, server and client accordingly.

**Implementation:**

* We used a channel of communication between the client as server as a medium in order to transfer the packets from the client to the server.
* Each packet has its own unique sequence that is decoded to 0 and 1.
* The main logic is that the channel is both a receiver and a sender/server.
* Receives packets from client and sends them to the server.
* Receives acknowledgements from server and sends them to the client.
* 2 ports necessary (client-channel, channel-server).
* Set initial conditions to send the first packet (acknowledgment received in client to begin intracommunication and send packets to server through the channel)
* When the acknowledgement for a specific packet is sent to the client, then the client continues (terminates the stop and wait) and prepares and sends the next packet.
* In the meantime, server stops and waits for next packet.
* Packet received from client to server, server’s stop and wait exits, sends acknowledgement.
* Acknowledgment received from server to client, client exits stop and wait send next packet and so on.
* Our implementation didn’t face any packet losses and therefore no timer was necessary to discard and resend the packets.

*\*Packets- frames*

**Execution Screenshot:**

These are the simultaneous print messages for the packets-frames and acknowledgements, in 3 different terminals running simultaneously, for the input: AAbC2

Text

Description automatically generated**Text

Description automatically generatedText

Description automatically generated**Client Channel Server