Pseudo Code

Tyler trepanier - A00850517

Client Server

Linux internal message queues

Table of Contents

[Client Server (main program entry) 2](#_Toc442041767)

[START 2](#_Toc442041768)

[PARSE COMMAND LINE 2](#_Toc442041769)

[Server 3](#_Toc442041770)

[SERVER 3](#_Toc442041771)

[SEARCH FOR CLIENTS 3](#_Toc442041772)

[CREATE NEW CLIENT PROCESS 3](#_Toc442041773)

[START (process) 4](#_Toc442041774)

[DESIGNATE PRIORITY 4](#_Toc442041775)

[OPEN REQUESTED FILE 4](#_Toc442041776)

[Open requested file 4](#_Toc442041777)

[SEND MESSAGES 4](#_Toc442041778)

[PACKETIZE DATA 5](#_Toc442041779)

[END (process) 5](#_Toc442041780)

[Client 7](#_Toc442041781)

[CLIENT 7](#_Toc442041782)

[OUTPUT FUNCTION 7](#_Toc442041783)

[DISPLAY MESSAGE 7](#_Toc442041784)

[PROMPT USER INPUT 8](#_Toc442041785)

[READ SERVER RESPONSE 8](#_Toc442041786)

# Client Server (main program entry)

Client Server is the combined entry point into the program. It will house the initial code that separates Client and Server functionality. When initially executing the program, you specify whether or not you are working as a client or you are working as the server.

## START

Goto PARSE COMMAND LINE

## PARSE COMMAND LINE

Read the command line arguments

IF command line specifies server

Goto SERVER

ELSE

Goto Client

# Server

This sections comes into effect when a user has selected the server option when running this application. The purpose of this section is to separate the functionality of the client and server functions. The Server’s responsibilities include creating the message queue where all clients will connect to. From there, the server will respond to all client file requests and return either error information or the file contents itself.

## SERVER

Create ClientServer msg queue

Assign Message-Type 1 to be the messages designated Client to Server

Goto SEARCH FOR CLIENTS

## SEARCH FOR CLIENTS

Check for any incoming messages from clients.

* Failure no new messages: RESTART SEARCH FOR CLIENTS

Pass message to CREATE NEW CLIENT PROCESS.

Goto CREATE NEW CLIENT PROCESS

## CREATE NEW CLIENT PROCESS

Check message data for client’s PID

Check message for client‘s priority

IF no priority specified, set priority to maximum

Fork a child process

IF Process is the child

Pass client information to this process.

Assign the task START (process)

Goto START (process)

ELSE

This is the Parent Process

Goto SEARCH FOR CLIENTS

## PROCESS CLIENT

Get the client’s message.

Prepare the series of messages with the client’s priority and their assigned message type.

Goto DESIGNATE PRIORITY.

Get the file’s name from received message.

Attempt to open the file.

* Success:
  + Pass the File, client’s message type and priority to PACKETIZEDATA.
  + Use PACKETIZEDATA to split the file’s contents and send them to the client.
  + Goto PACKETIZE DATA.
* Failure:
  + Send a message using the client’s specified message type back to the client indicating that there was an error with their file request.

The message(s) have been sent successfully.

Terminate this process.

## DESIGNATE PRIORITY

Get Client’s Priority and PID

Designate new length of message to be MAXSIZE / PRIORITY

Goto OPEN REQUESTED FILE

## OPEN REQUESTED FILE

Get message data from client

Read desired Filename from message data

Open requested file

* Failure: File cannot be opened for reading
  + Send “Error: Cannot open file” message to SEND MESSAGE
  + Goto SEND MESSAGE
* Success: File has been read
  + Send this file to PACKETIZE DATA
  + Goto PACKETIZE DATA

## SEND MESSAGES

Grab assigned PID to indicate message type (destination of message)

Grab the message queue

Grab the message

Grab the “more” flag to indicate more messages

Send message to the message queue using the message type as its address.

IF “more” flag has been specified

Return to PACKETIZE DATA (restart its file read loop)

Goto PACKETIZE DATA

## PACKETIZE DATA

Grab file contents

Grab the message queue

Grab the PID

Grab client priority

Read file until designated length of message

WHILE the file can be read

Check if file can be read

IF file cannot be read

Get out of while loop

Send the portion of the message to SEND MESSAGES

Send the PID to SEND MESSAGES

Send the Message Queue to SEND MESSAGES

Send the “more” flag to SEND MESSAGES

Goto SEND MESSAGES

Restart loop

File has finished reading

Goto END (process)

## END (process)

Free all used resources.

Terminate this child process.

# Client

This sections comes into effect when a user has selected the client option when running this application. The purpose of this section is to separate the functionality of the client and server functions. The Client is responsible for reading file requests, made by the user from Standard Input, and requesting those files from the existing server. These requests will be made by taking advantage of the Linux message queue to allow for interprocess communication.

## CLIENT

Open pre-existing Clientserver message queue.

Prompt for user input.

Goto USERINPUT

Arrange the user request into the format of “<filename> <(if available)priority> <client ID>”

Put the arranged user request into a message.

Send the message to the ClientServer message queue for the server to read.

Read from the Server to all clients message queue on the server’s response.

Goto READ SERVER RESPONSE.

Assign the task of reading to the DisplayMessage which will create a new thread.

Goto DISPLAYMESSAGE.

## OUTPUT FUNCTION

Get the message read from the server’s response.

Display the message to Standard Output.

Terminate the thread.

## DISPLAY MESSAGE

Get the message read from the server’s response.

Create a detached thread.

Pass the message to the detached thread so it can be read.

Start the thread.

Return to CLIENT.

## PROMPT USER INPUT

Prompt the user for the requested file in the format of “<filename> <(optional) priority>”

* Failure: Tell the user the format of the message and keep trying to read until the user’s input matches the requested format.

Return to CLIENT with the user’s accepted input.

## READ SERVER RESPONSE

In a continuous loop that will end ONLY when the entire file’s contents are read.

Get the server’s message from the message queue using the assigned message type.

Read the message into a temporary message.

Display the received message’s contents.

Goto DISPLAY MESSAGE.

Search the message for the EOF character to see if there’s no more messages to be received.

* + EOF found: Exit the continuous loop.
  + More messages: Restart the continuous loop and read more messages.

Release all used resources.

Return to CLIENT.