**Operating Systems Project 4**

**University at Albany Department of Computer Science**

**ICSI 500**

Table of Content

1.Introduction

2. Background

3. System documentation

i. A high-level data flow diagram for the system ii.

A list of routines and their brief descriptions

4. Implementation details

5. [5%] Test documentation

i. How you tested your program

ii. Testing outputs e. [5%] User documentation

1. Where is your source
2. ii. How to run your program
3. iii. Describe parameter (if any)

1.INTRODUCTION

2.BACKGROUND PROBLEM

You are to develop a data processing system to process strings of characters. The solution must replace all instances of lowercase vowels with uppercase ones and append to the last set of strings the sum of all numbers found. You are to design and implement a client/server application where the server will respond to different client requests by creating two dedicated new processes. One process will read from the socket, decode the received messages, and will create seven threads. The other process will encode the message processed by the threads and will write the resulting data to the socket. All communications between both the client and the server will be encoded according to the format defined in the project 3 format

3. SYSTEM DOCUMENTATION

1. A high-level data flow diagram for the system ii.

THE SERVER

Accepts multiple client requests and for each request creates two child processes. These processes are to be named serverDecoder and ServerEncoder. The ServerDecoder will read from the socket, decode the received messages, and will create the following seven threads: 1 The charA thread will read data provided by the decoder process and will replace all lowercase a character with uppercase. It will share the data received with the charE thread through a queue of messages.

CSI 500 Operating Systems Spring 2022

2 The charE thread component will scan the data shared and replace all lowercase b character with uppercase E. It will then share the modified data with the charI thread through another queue of messages.

3 The charI thread component will scan the data shared and replace all lowercase i character with uppercase I. It will then share the modified data with the charO thread through another queue of messages.

4 The charO thread component will scan the data shared and replace all lowercase o character with uppercase O. It will then share the modified data with the charU thread through another queue of messages.

5 The charU thread component will scan the data shared and replace all lowercase u character with uppercase U. It will then share the modified data with the digit thread through another queue of messages.

6 The digit thread component will scan the data shared and computes the sum of all digits found. I will then share the processed data with the writer thread through another queue of messages.

7 The writer thread will share the data received with the serverEncoder process. The serverEncoder will encode the resulting data and will share it with the client through the socket connection.

A list of routines and their brief descriptions

4. IMPLEMENTATION DETAILS

Data follow through out the system

ServerGen

User

User

Server

filename.binf

filename.inpf

Client

Filename.outf

The following was implemented in c++.

Transmitter

* Application layer
  + transApp()
    - reads the input
    - for the number of necessary blocks
      * calls transDataLink.addSynAndControl to format the control characters
      * calls tranPhysical.convertCharToBinary for every character in the input block
* Data Link layer
  + class transDataLink
    - void addSynAndControl(int n)
      * calls tranPhysical.convertIntToBinary() twice with 22 and once with the length of the input as parameters
* Physical layer
  + class tranPhysical
    - int \* convertIntToBinary(int n)
      * converts decimal integer to binary and adds odd parity
      * puts results in output file and returns the resulting integer as well
    - int \* convertCharToBinary(char s)
      * performs the same action as convertIntToBinary but with characters as input

Receiver

* Application layer
  + recApp::recApp()
    - reads transmitter output file
      * for each block
        + calls recDataLink.addSynAndControl with the current block number as input
        + for each byte in the block

calls

recPhysical.getByte() with the bytes position as input and the byte as output

recPhysical.convertBinaryToChar() with the byte as output

writes to output

* Data link layer
  + class recDataLin
    - int addSynAndControl(int block)
      * checks the control characters are correct
      * returns the number of characters specified by the third control character
* Physical layer
  + class recPhysical
    - int convertBinaryToDecInt(string revNum)
      * converts binary to integer
      * removes and checks odd parity
    - string getByte(int startPos)
      * returns a string which represent a byte starting at start position
    - convertBinaryToChar(string byte)
      * converts a byte to a character

Error Generator

* ErrorGen(int n)
  + reads in the output of transmitter
  + adds n errors to the file making sure to have no collisions
  + reports the frame byte and bit location of errors to the console

RuName

* runs all three files
* comment error gen to remove errors from transmission

**Test documentation**

Testing was preformed using various input both with error generation and no error generation.

Example input

“aracters and have the .inpf file extension. The data to be transmitted must be structured as a set of blocks. Each block will consist of 2 SYN characters, ASCII 22, one control character to indicate (zeros and ones) of data block followed by a maximum of 64 data characters. Assume that there are no trailing control characters. Every block transmitted musst contain 64 data characters except possibly the case where the remainder of the file cannot fill the buffer. Each character will consist of 7 information bits and”

Example output with errors turned off

“aracters and have the .inpf file extension. The data to be transmitted must be structured as a set of blocks. Each block will consist of 2 SYN characters, ASCII 22, one control character to indicate (zeros and ones) of data block followed by a maximum of 64 data characters. Assume that there are no trailing control characters. Every block transmitted must contain 64 data characters except possibly the case where the remainder of the file cannot fill the buffer. Each character will consist of 7 information bits and”

Example output with errors turned on (100 errors)

“araCters qnd have the >iorf fi|d Extensinn. The data to be translitted!Must be structupee as a set of blocks# Each block`will sonsist of 2 SYn!chaR!cters, ASCII 22, o~e0cnurom!charaCtgr vo in”

**User documentation**

Source code is the attached files, in the src folder. The majority of code are in the .cpp files with .h files making the classes accessible by other .cpp.

To run the code run RunName.cpp. Change the input in ErrorGen to add errors. This is also the only parameter.

5. TEST DOCUMENTATION

i. How you tested your program

ii. Testing outputs e. [5%] User documentation

1. Where is your source
2. ii. How to run your program
3. iii. Describe parameter (if any)