COSC 3360—Fundamentals of Operating Systems
Assignment #2 A Simple File Repository
Due Wednesday, March 28 at 11:59 PM

Objective

You are to implement a simple file transfer protocol for a file repository. Your server should be a stand-alone process that continuously awaits requests from other processes (the clients). All communications between the server and its clients should be done through Internet domain sockets.

Overview

You are to write two programs:

1. A client program that will connect with your server and request file downloads. You should name it **`FirstName_LastName_client.cpp`** or **`FirstName_LastName_client.c`**.
2. A server program that will wait for connection requests from its clients and fulfill the requests of its clients. You should similarly name it either **`FirstName_LastName_server.cpp`** or **`FirstName_LastName_server.c`**.

The client program

Your program should prompt for input from the user terminal and recognize three commands:

1. **`get filename`**, which will request the contents of a file from the server home directory and store them in a file with the same name.
2. **`exit`**, which will terminate the client, send an exit message to the server, and close the connection.
3. **`terminate`**, which will terminate the client and the server.

After each transfer, your client should print out the name of the file and the number of bytes being transferred. It ***could*** be something like:

```
Received file test.txt (256 bytes)
```

You do not have to worry about overwriting existing files (just do it!) or handling pathnames. The only error condition you will have to report is when the requested file does not exist.

The server program

Your server will be another process on your PC or your Mac: we will just pretend they are on different machines. All server filenames will refer to a specific directory called **`Repository`**. Your server should do an endless loop waiting for a message from your client:

1. If the message is a **`get`** request, the server should print out the name of the requested file and either send the contents of the specified file to the client or report that the file is missing. When it is done, your client should print out the number of bytes being transferred. It ***could*** be something like:

   ```
   A client requested the file test.txt
   Sent 256 bytes
   ```

   or

   ```
   A client requested the file xyz.txt
   That file is missing!
   ```

2. If the message is a **`terminate`** request, the server should print out **`Goodbye!`** and terminate immediately: this is not what a real server would do, but it will simplify everyone's life.

Your server should fork a separate child for each client to be able to handle several requests in parallel.

Hints

1. Please refer to the two online socket tutorials at:

   http://www.linuxhowtos.org/C_C++/socket.htm
   http://www.cs.uh.edu/~paris/3360/Sockets.html

   or through the course Team. You can include any code from these two documents in your submissions.
2. Look also at the sample program pair to be posted on Teams.
3. Your server and client processes will read messages byte by byte and have no way to know message lengths. You may have to prefix server replies with a fixed-format integer specifying the length of each horoscope. Both **`sprintf()`** and **`sscanf()`** could come in handy.
4. You do not have to worry about zombies and can safely ignore the **`fireman()`** call in the primer.

These specifications were updated last **Wednesday, February 28, 2024**. Check for updates, corrections, and extensions on Teams or Prulu.