CSCI 4210 Operating System
Fall, 2000
Project 3
Due on 10/17/2000

Write a version of the `ftp` client and server which follows the following protocol. Note that any client should be able to communicate with any server.

The client can send the following requests to the server:

`GET` *filename* - a request for the server to send a file to the client

`DIR` - a request for the server to send the names of all files in the current directory to the client.

`PRT` *port number* - a request for the server to open a data connection to that port on the client.

`BYE` - a request to terminate the connection.

Once a connection has been established, the first command that the client should send to the server is a `PRT` command. This means that the client is listening on that port and the server should connect to it. The original channel (the command channel) will be used to exchange client requests and their replies. The second channel that was opened in response to the `PRT` command (the data channel) will be used to exchange data. Both should stay open until the client sends a `BYE` request.

For all of these messages except `BYE`, the server responds with either an `OK` or an `ER`. An `OK` reply means that it can handle the request. The server should not respond to a `PRT` command until it has successfully connected to the port.

Once the server has sent an `OK` to a `DIR` or a `GET` request, it will start sending the requested data on the data channel. The format is that it will send the number of bytes to be sent in ascii form, followed by a single space followed by the data. For example, if it were sending a 234 byte file, it would write a total of 238 bytes, the digit '2', the digit '3', the digit '4', a space, and then the file.

Since the server has to know how many bytes to send in response to a `DIR` request, it should write the output to a temporary file first.

All messages on the command port in both directions should be terminated by a newline (ascii 10). Messages on the data transfer channel should not be terminated by a newline, although if the last character of the file is a newline, as it often it, this should be sent.

In the event of an error condition, the server sends an `ER` message followed by a single space, followed by a brief error message. This should also be terminated by a newline.

The client does not need to respond to error messages other than to display them to the user.

The client should take two arguments, the name of the host on which the server is running and the port number. The client should have simple command line interface similar to the real ftp.

As always, your program should be able to handle a wide variety of error conditions. However, both the client and the server can assume that the stated number of bytes to be sent is correct (It is very difficult write code to deal with this type of error.).

For 5 points extra point credit:

The server should be able to handle multiple simultaneous clients. When it is invoked, the user should pass the port name in as an argument. The server should run as a daemon; when called, it should disconnect from the shell and put itself in background. This means that it should stay running even after the user logs out (However, because of security features of RCS, it will not be able to access any files.). Your server should not allow zombie processes to hang around.

Alert: After you are through testing your server, make sure that you explicitly kill it.