Requirement

You are required to write a server program and a client program using the system calls in UDP socket programming only. The server address and port number are known to the client. On the other hand, the server does not know the client address in advance. The client address is obtained by the server when it receives a request from a client.

Client: provides an interface that repeatedly asks the user to enter a request and sends the request to the server. Each reply or error message returned from the server should be printed on the screen. The interface provided by the client should include an option for the user to terminate the client.

Server: repeatedly receives requests from the client, performs the service and replies to the client. The received requests and the produced replies should be printed on the screen. The files available for remote access must be stored on the local disk of the server computer.

- ## Read content of a file(Easy)

  Description: A service that allows a user to read the content of a file

  Parameter: pathname, offset(in bytes), the number of bytes to read from the file

  Return: the given number of bytes of the file content starting from the designated offset in the file (the offset of the first byte in the file is 0)

  Error message should be returned if the file does not exist on the server or if the offset exceeds the file length

- ## Insert content into a file(Easy)

  Description: The service inserts the sequence of bytes into the file at the designated offset in the file. The original content of the file after the offset is pushed forward.

  Parameter: pathname, offset (in bytes), a sequence of bytes to write into the file

  Return: an acknowledgement on successful insertion

  Error message should be returned if the file does not exist on the server or if the offset exceeds the file length

- ## Monitor updates

  Description: A service that allows a user to monitor updates made to the content of a specified file at the server for a designated time period called monitor interval. After registration, the Internet address and the port number of the client are recorded by the server. During the monitoring interval, every time an update is made by any client to the content of the file, the updated file content is sent by the server to the registered client(s) through callback. After the expiration of the monitor interval, the client record is removed from the server which will no longer deliver the file content to the client

  Parameter: pathname, the length of monitor interval

  Return: the updated file content

- ## One idempotent operation
- ## One non-idempotent operation
- ## Client-side caching

  Description: The file content read by the client is retained in a buffer of the client program. If the file content requested by the client is available in the cache, the client may read the cached content directly without contacting the server. On the other hand, the updates made by the client to file content are always sent to the server immediately. The approximate one-copy update semantics (used by NFS as described in the lecture) is required to be implemented to maintain cache consistency.

  Parameter: freshness interval t

  Need to maintain some meta-information of cached content (e.g., the file pathname of the cached content, the time when the cached content was last validated).

- ## Request/reply message formats, fully implement marshaling/unmarshalling(Implemented)

  Different operations may need different types of arguments and return different types of results. All messages transmitted between the server and the clients must be in the form of a sequence of bytes. you must marshal the integer values, strings etc. before transmission and unmarshal them upon receipt.

- ## Two different invocation semantics: at-least-once and at-most-once

  Need to implement techniques like timeouts, filtering duplicate request messages, and maintaining histories.

  Which semantics to use may be specified as an argument in the command that starts the client/server.

  You can assign a request identifier to each request for detecting duplicate requests.

  Need to design experiments to compare the two invocation semantics. Show that at-least-once invocation semantics can lead to wrong results for non-idempotent operations, while at-most-once invocation semantics work correctly for all operations.

- ## Simulate the loss of request and reply messages