CMU 15-640 Project 1
Design Document

Xiaoyun Ye
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`xye2@andrew.cmu.edu`

System description

This is an RPC system to allow remote file operations (open, read, write, . . .). It includes a server process and a client stub library. The server provides the remote file services: it lets a remote client read, modify, and delete files in the file space. The client stub library can perform RPCs.

The system can handle the following standard C library calls: `open`, `close`, `read`, `write`, `lseek`, `stat`, `unlink`, `getdirentries`, and non-standard `getdirtree` and `freedirtree` calls.

The system has its own protocol and messaging standard for the messages between client and server.

Messaging protocol between client and server

To execute each function call, the client sends a message to the server. And after executing, the server replies with the error and result information to the client.

The messages client sends follow a uniform format: the first 4 bytes of the message is the message's size. The following bytes contain related information varying according to the specific function call.
The system also assigns an id for each type of function call for easy identification for the server when receiving a client's request. `open`=1, `close`=2, `write`=3, `read`=4, `lseek`=5, `__xstat`=7, `unlink`=8, `getdirentries`=9, `getdirtree`=10.

Message from client to server

The client's message content for each function call is as follows:

`open`: message length, function type id, flag, path name size, mode, path name.
`close`: message length, function type id, file descriptor.
`read`: message length, function type id, file descriptor, count.
`write`: message length, function type id, file descriptor, count, buf.
`lseek`: message length, function type id, whence, offset.
`__xstat`: message length, function type id, version, path name size, path name.
`unlink`: message length, function type id, path name size, path name.
`getdirentries`: message length, function type id, file descriptor, maximum bytes, base pointer.
`getdirtree`: message length, function type id, path name size, path name.

Message from server to client

The server's reply message info after executing each function call is as follows:

`open`: file descriptor, errno info.
`close`: errno info.
`read`: errno info, # of bytes, content read.
`write`: errno info, # of bytes written.
`lseek`: errno info and result info.
`__xstat`: errno info, result info, buf.
`unlink`: errno info, result info.
`getdirentries`: errno info, result, # of bytes.

Implementation of getdirtree function call

The system applies DFS function to transverse the whole directory when executing the `getdirentries` function call. After obtaining the sub directories of the parent directory, Recursively apply DFS to each of the sub directory.