Yun Chen Sheng

15640- Spring 2019

Project 1: Transparent Remote File Operations

Document:

The serialization protocol between server and client:

The protocol is as follow, after the function call, the client will connect to server and create a TCP connection for this call only, the life time of TCP connection only exist until the function return, and it would sent the function name (for example, an open function would sent OPEN and a space to server), the input argument(separated by space), and finally end with two newline char ("\n\n"). The server side TCP connection would receive this package and use strncmp() to check the first several character in package. If it finds match, it would call the corresponds package process function to process the package. The package process function is list in different .c file called package_process.c, and each will create a local buffer and copy the package into its local buffer one parameter a time. The package process function first search for space using strchr() and mark it as indexStart, then it search for another space and mark as indexEnd. It copy the content between indexStart and indexEnd to the local buffer and then save to local variable with correct type. Finally, for the last one parameter, it would search for "\n" character, since we put "\n\n" at the end of package at client side ("\n\n" is to avoid any potential off one count problem.) Notice that evert parameter is treated the same, the int and the char* are both separated by space.

When the remote call return, we get the return value, errno, and return payload(char*), the return package is as follow:

RETURN:{return value} ERRNO:{errno} {payload} (with payload)
RETURN:{return value} ERRNO:{errno} (without payload)

Every part of return package is separated by space, and the return value always indicate the payload size if payload exist. Notice that we know if we will receive payload or not which is depends on function type (ex: open() would not have payload), and we also know the size of payload by reading the return value first, so we can memcpy() the payload to local buffer with correct size. These functions are handle by read_msg() and read_msg_and_payload() in mylib.c. An important note is that we should always check if return value is -1 at both server side and client side. At server side, if return value is -1, we should set payload siez to 0, and we should not do memcpy(). At client side, if we see return value is -1, we should not read payload.