Encode protocol for client and server					
	Open	Close	Read	Write	Stat
Client Encode	Request length Opcode Flags Mode Path	Request length Opcode fd	Request length Opcode Fd Read length	Request length Opcode Fd write length write buf	Request length Opcode Version Path length Path
Server Encode	Response length Return Opcode Errno	Response length Return Opcode Errno	Response length Return Opcode Errno Content	Response length Return Opcode Errno	Response length Return Opcode Errno Stat-buf content

	unlink	getdirentries	getdirtree	lseek
Client Encode	Request length Opcode Flags Mode Path	Request length Opcode Fd Nbytes Basep	Request length Opcode Path length Path	Request length Opcode Offset Whence
Server Encode	Response length Return Opcode Errno	Response length Return Opcode Errno Content Basep	Response length Return Opcode Errno Struct serialized body	Response length Return Opcode Errno

Some tips for this project:

1. Time out issue:
   In order to solve the time-out problem, we need to use the while loop to receive the message small chunk by small chunk instead of the total large message. I will use the received-length as the pointer for the response message to record where I should concatenate new received information to the response.
2. Not your file case:
   For this case, I use a range of file descriptor for the RPC (>1024), and a range of file descriptor as the local ones.
3. BadbadFile:
   After several tests, I figured out that I need to set up the fd pool to store the fd that we create yet. Every time we need to check the fd, whether it is in the pool. In this way we could totally avoid the bad fd that was not create by my RPC.
4. Pointer Issue:
   In this project, we need to pay attention to the pointer offset we used to store the place information, It will influence the pack and unpack accuracy.