|  |  |  |  |
| --- | --- | --- | --- |
| open | close | write | read |
| client -> server  int size  int opcode - 0  int flag  mode\_t mode  string path  server -> client  int size  int opcode - 0  int return  int errno | client -> server  int size  int opcode - 1  int fd  server -> client  int size  int opcode - 1  int return  int errno | client -> server  int size  int opcode - 2  int fd  size\_t count  string buf  server -> client  int size  int opcode - 2  ssize\_t return  int errno | client -> server  int size  int opcode - 3  int fd  size\_t count  server -> client  int size  int opcode - 3  ssize\_t return value  int errno  string buf |
| lseek | \_\_xstat | unlink | getdirentries |
| client -> server  int size  int opcode - 4  int fildes  off\_t offset  int whence  server -> client  int size  int opcode - 4  off\_t return  int errno | client -> server  int size  int opcode - 5  int var  struct stat  string path  server -> client  int size  int opcode - 5  int return  int error | client -> server  int size  int opcode - 6  string pathname  server -> client  int size  int opcode - 6  int return  int errno | client -> server  int size  int opcode - 7  int fd  size\_t nbytes  off\_t basep  server -> client  int size  int opcode - 7  ssize\_t return  int errno |
| getdirtree | freedirtree |  |  |
| client -> server  int size  int opcode - 8  string path  server -> client  size\_t size  int opcode - 8  int success  int errno  string path  int number  string path  int number  ... ...  ... ... | client -> server  int size  int opcode 9  string path  int number  string path  int number  ... ...  ... ...  server -> client  int size  int opcode 9  int success  int errno |  |  |

How to avoid timeout caused by recv() and send()

My implementation: the first four bytes of request and response is an int, which represent the length of this request/response. I check return value of each recv() and send() and use loop in order to receive the exact value.

How to discriminate local file operation and remote file operation

For remote file operation, all fd will be added an offset of 512. So for fd within[0, 512], it is considered as local file. For fd within [512, 1024], it will considered as remote file.

How to deal with bad file descripter

For remote file fd, I use a fd\_set to store fd info. When an fd is open or close, I use FD\_SET and FD\_CLR to update its state. When a client tries to operate on invalid file descripter. It will return -1 and set errno directly.