Portocol

```
enum SystemCall {
  OPEN,
  CLOSE,
  READ,
  WRITE,
  LSEEK,
  STAT,
  XSTAT,
  UNLINK,
  GET_DIR_ENTRIES,
  GET_DIR_TREE,
};
```

For the communication between Client and Server. Client first send a int32_t to tell server, which system call that client requesting, It's defined in “util.h”.

After that, sever function is also made to handle recv/send between server and client.

```
bool send_exact(int fd, const void* buf, int size);
bool recv_exact(int fd, void* buf, int size);
bool send_int(int fd, int32_t i);
bool send_int64(int fd, int64_t i);
bool send_string(int fd, const char* str);
bool recv_int(int fd, int32_t* i);
bool recv_int64(int fd, int64_t* i);
bool recv_string(int fd, char* str);
bool send_dirtree(int fd, const char* path);
bool recv_dirtree(int fd, struct dirtreenode** dptr);
```

All these function is actually handled by send_exact(), recv_exact(), which take charge of how many bytes actually needs to be processed. They all accept size and while(size > 0) it keeps reading/writing until return value <= 0 which ensure they get all the data done.

For the errno,I simply pass them to client which means I send it every time whenever it success or not. The client recv them every time as well.

```
fd_set* opened_files = malloc(sizeof(fd_set));
```

However, I found when running test, my code still gets problems like bad fd. So I use FD_SET to monitor every fd at server and manual add 2048 to ensure it will not be negative. The client the decide what to do based on the return val. At the mean time, every return val (not only open) must be checked.

```
Status pipeline_exact(int dstfd, int srcfd, int
len) {
  char buff[BUF_LEN];
  while (len > 0) {
    int actual = min(len, BUF_LEN);
    if (!read_exact(srcfd, buff, actual)) {
      return SRC_ERROR;
    }
    if (!write_exact(dstfd, buff, actual)) {
      return DST_ERROR;
    }
    len -= actual;
  }
  return OK;
}
```

To ensure the big data transfer, I made a function pipeline_exact(), which directly reads from socket and write into fd. But it will still gets TIME OUT. So I had to setsockopt to TCP_NODELAY which solve the problem.