# Project 3 Technical Document

CSE521 Introduction to Operating System

**Wei Zhu**
**12/15/2010**

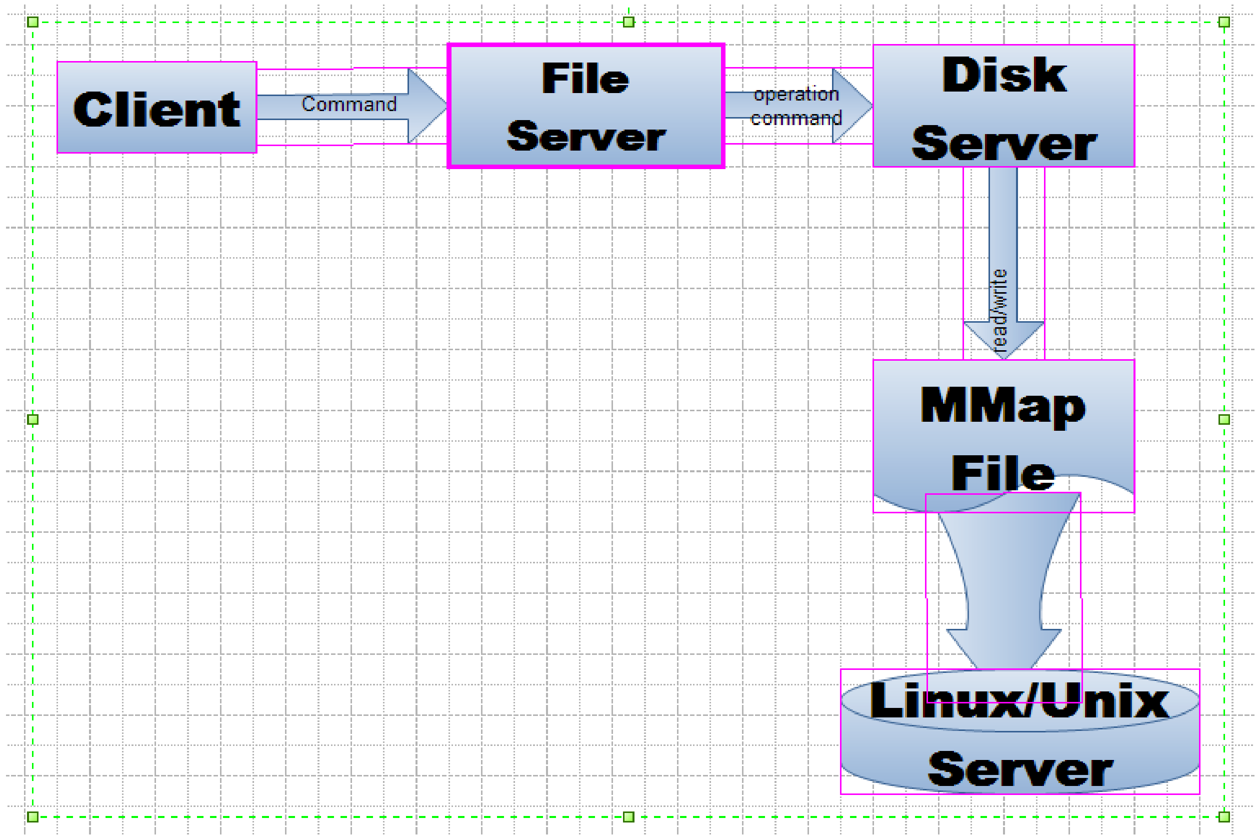
## Introduction

This document is written for the final project of CSE521 Introduction of Operating System.

This project implements a simple file system:

1. Disk storage system
2. File system
3. User manual command
4. Disk/File Client

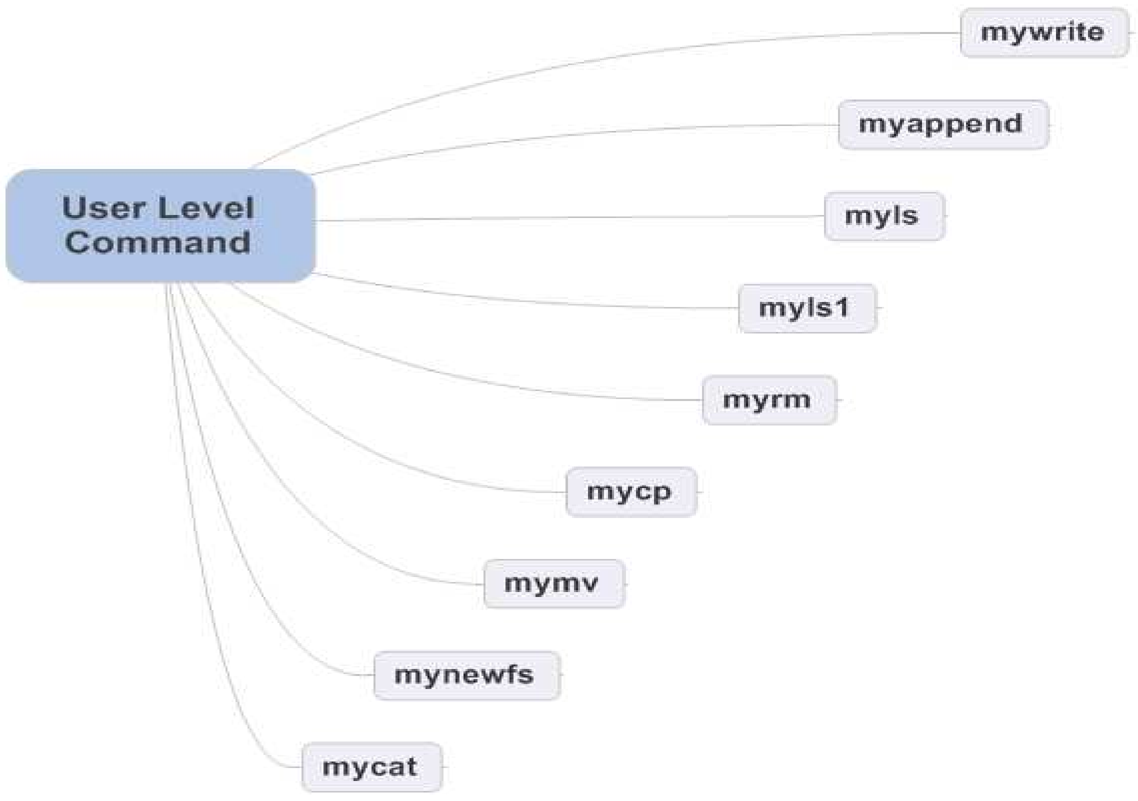
## The Architecture of the System:



# Users manual

## Here are a few basic command-line user level commands.

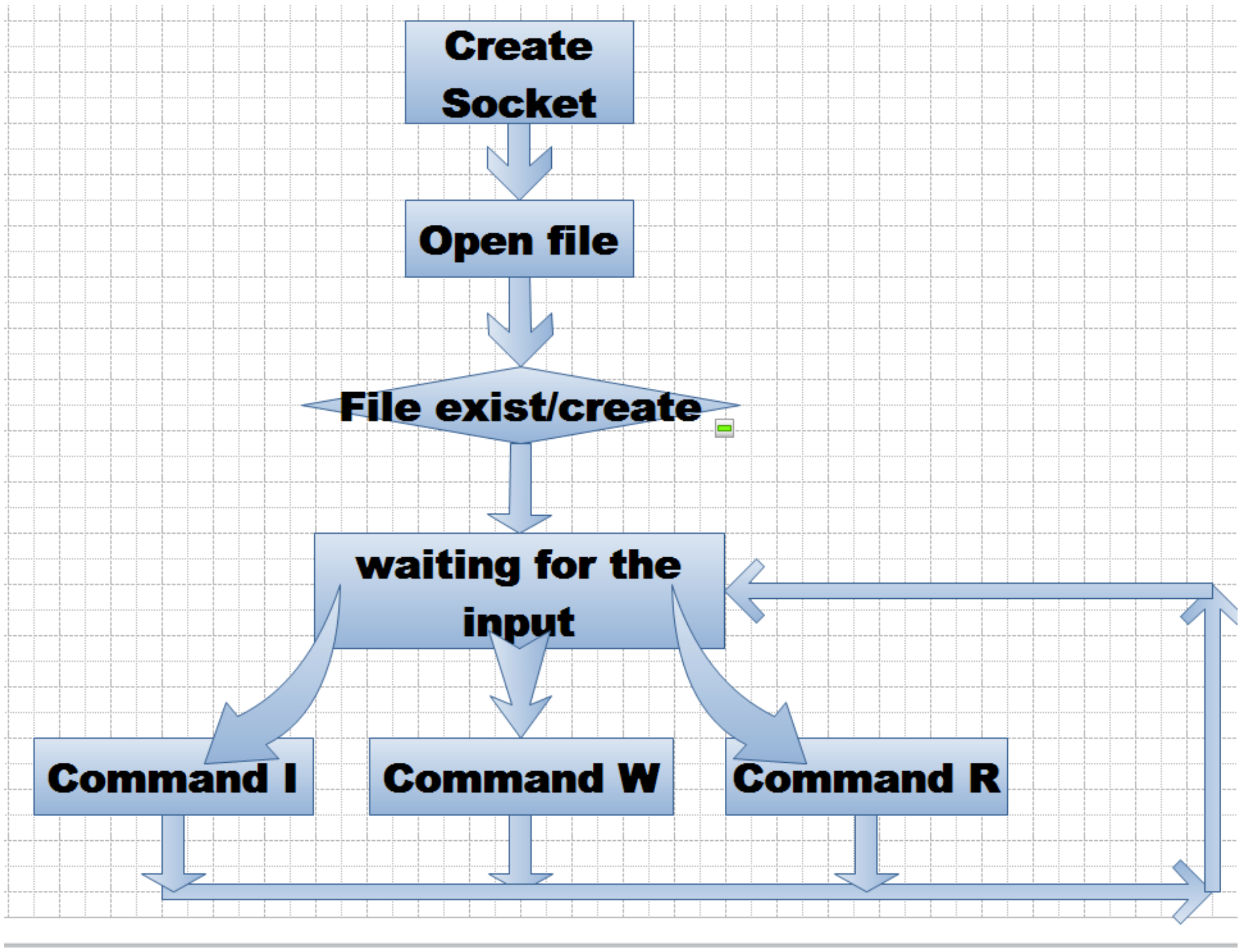
1. **mycat** *filename*: show the contents of filename.
2. **mywrite** *filename*: Read stdin and overwrite filename with all the data from stdin.
3. **myappend** *filename*: Append all the data from stdin to filename.
4. **myls**: show the list of files in the directory.
5. **mylsl**: show the long form list of files in the directory.
6. **myrm** *filename*: remove filename.
7. **mycp** *filename1 filename2*: copy filename1 to filename2 (overwriting filename2 if it already exists.
8. **mymv** *filename1 filename2*: rename filename1 to filiename2 (overwriting filename2 if it already exists).
9. **mynewfs** : format the file system, removing all contents

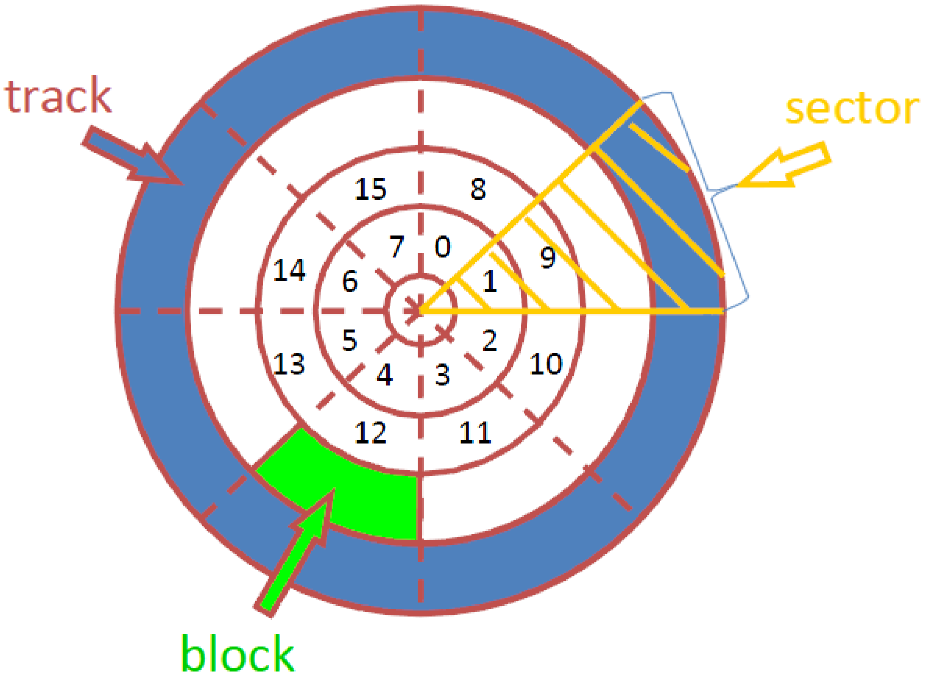


# Sub-System Architecture

## Disk-Storage Server

Basic Disk storage server is a Unix-domain socket server to simulate a physical disk. The simulated disk is organized by cylinder and sector. Let the number of cylinders and number of sectors per cylinder is command line parameters. Assume the block size is fixed at 256 bytes. Your simulation should store the actual date in a real disk file. So you'll want a filename for this file as another command line option.

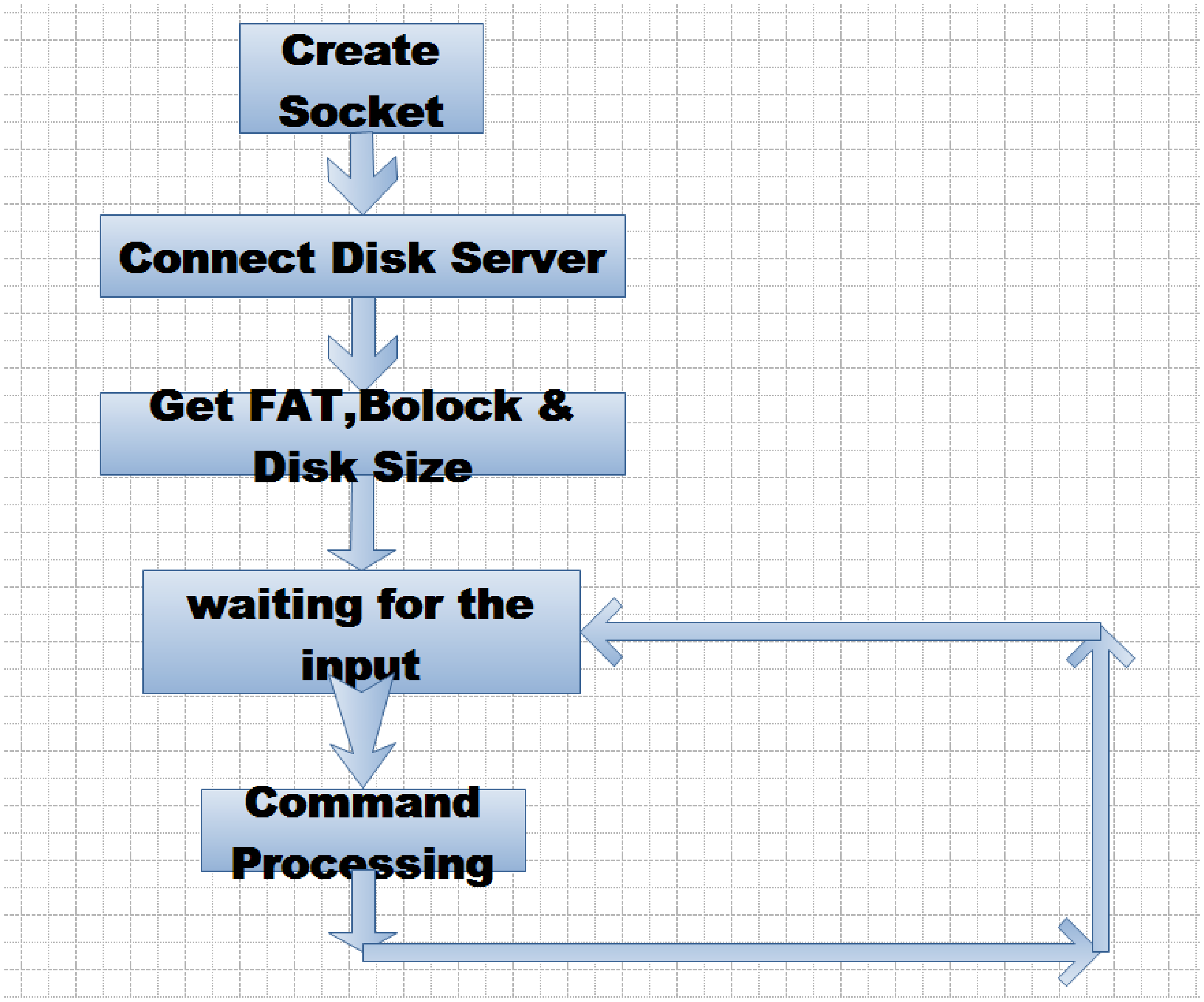




"Tracks" – Each track is defined to be a certain distance away from the middle spindle
"Sectors" – The pie slices of the disk that intersects the tracks
"Blocks" – The intersection of a track and a sector

## File System Server

Implement a flat file system that keeps track of files in a single directory (table). The file system should provide operations such as: initialize the file system, create a file, read the data from a file, write a file with given data, append data to a file, and remove a file, etc…



The Following command was supported:
C:
Create a new file and update the file table;
If it already exists, fail;

R:
Read the file content;

W:
Write to specific file and update the file table;

F:
Format the whole file system and clear the entire file table

# The Interface Definition

## User Client vs. File Server

**I**: information request. The disk returns two integers representing the disk geometry: the number of cylinders, and the number of sectors per cylinder.

**R *c s***: read request for the contents of cylinder *c* sector *s*. The disk returns '1' followed by those 256 bytes of information, or '0' if no such block exists. (This will return whatever data happens to be on the disk in a given sector, even if nothing has ever been explicitly written there before.)

**W *c s l data***: write request for cylinder *c* sector *s*. *l* is the number of bytes being provided, with a maximum of 256. The data is those *l* bytes of data. The disk returns '1' to the client if it is a valid write request (legal values of *c*, *s* and *l*), or returns a '0' otherwise.

## The File Server vs. Disk Server

**F**: format. Format the file system on the disk, by initializing any/all of tables that the file system relies on.

**C *f***: create file. This will create a file named f in the file system. Possible return codes: 0 = successfully created the file; 1 = a file of this name already existed; 2 = some other failure (such as no space left, etc.).

**D *f***: delete file. This will delete the file named *f* from the file system. Possible return codes: 0 = successfully deleted the file; 1 = a file of this name did not exist; 2 = some other failure.

**L *b***: directory listing. This will return a listing of the files in the file systems. *b* is a Boolean flag: if '0' it lists just the names of all the files, one per line; if '1' it includes other information about each file, such as file length, plus anything else your file system might store about it.

**R *f***: read file. This will read the entire contents of the file named *f*, and return the data that came from it. The message sent back to the client is, in order: a return code, the number of bytes in the file (in ASCII), a white-space and finally the data from the file. Possible return codes: 0 = successfully read file; 1 = no such filename exists; 2 = some other failure.

**W *f l data***: write file. This will overwrite the contests of the file named *f* with the *l* bytes of data. If the new data is longer than the data previous in the file, the file will be made longer. If the new data is shorter than the data previously in the file, the file will be truncated to the new length. A return code is sent back to the client. Possible return codes: 0 = successfully written file; 1 = no such filename exists; 2 = some other failure (such as no space left, etc.).

**A *f l data***: append to file. This will append the *l* bytes of data to the end of the current contents of the file named *f*. A return code is sent back to the client. Possible return codes: 0 =

successfully appended to file; 1 = no such filename exists; 2 = some other failure (such as no space left, etc.).

# Testing Result

## The Disk-Storage Server

```
. /stdindclient MYDS

initial the parameter

created socket

loop entered

I

command line is I

send line is I

read count is 4

received response: 5 10

loop entered

W 1 1 3 abc

command line is W 1 1 3 abc

send line is W 1 1 3 abc

read count is 1

received response: 1

loop entered

R 1 1

command line is R 1 1

send line is R 1 1

read count is 257

received response: 1abc

loop entered
```

## The File System Server

```
>C file1

command line is C file1

send line is C file1

read count is 1

received response: 0

>W file1 abc

command line is W file1 abc

send line is W file1 abc

read count is 21

received response: Unsupported command

>W file1 4 abc

command line is W file1 4 abc

send line is W file1 4 abc

read count is 1

received response: 0

>A file1 4 efg

command line is A file1 4 efg

send line is A file1 4 efg

read count is 1

received response: 0
```

```
>R file1

command line is R file1

send line is R file1

read count is 6

received response: 04 abc

>W file1 4 abcd

command line is W file1 4 abcd

send line is W file1 4 abcd

read count is 1

received response: 0

>R file1

command line is R file1

send line is R file1

read count is 10

received response: 04 abcdefg

>W file1 4 abcd

command line is W file1 4 abcd

send line is W file1 4 abcd

read count is 1

received response: 0

> W file1 8 abcdefgh

command line is W file1 8 abcdefgh

send line is W file1 8 abcdefgh

read count is 1

received response: 0
```

## The User Manual Testing Script

```
. /mynewfs

echo
0123456789101112131415161718192021222324252627282930313233343536373839404142434445 46
4748495051525354555657585960616263646566676869707172737475767778798081828384858687
888990919293949596979899012345678910111213141516171819202122232425262728293031323 33
4353637383940414243444546474849505152535455565758596061626364656667686970717273747
5767778798081828384858687888990919293949596979899|./mywrite f1

echo
0123456789101112131415161718192021222324252627282930313233343536373839404142434445 46
4748495051525354555657585960616263646566676869707172737475767778798081828384858687
888990919293949596979899012345678910111213141516171819202122232425262728293031323 33
4353637383940414243444546474849505152535455565758596061626364656667686970717273747
5767778798081828384858687888990919293949596979899|./myappend f1

. /myls

. /mycp f1 f2

. /mycp f2 f3

. /mycp f3 f4

. /myrm f2

. /myrm f3

. /mymv f4 f5

. /mymv f5 f6

echo TA|. /myappend f6

. /myls1

. /mycat f6

echo 123|. /mywrite f9

echo 345|. /myappend f9

echo 678|. /myappend f9

echo 999|. /myappend f9
```