Contributors:

Abdullah Alshebly

Montague La France

**Technical Report**

**User Manual:**

Each program can be compiled using make-file it will generate both client and server executables. Both the client and server are need to be executed in separate terminal windows. And both the client and server should be executed concurrently, but server must be executed first so that client can run without giving connection failure message.

Each program server can accommodate 3 clients at a time and if a 4th client wants to join then server send a denial of service message to the Client. So, 4th client is not connected to server. A client can disconnect from the server using DISCONNECT message and that client is disconnected from server when server get this message. Moreover, when we have 3 Clients connected and 1 Client disconnect then the 4th client can join the server and there will be not denial of service because total at a time connection will remain 3.

**Program 1:**

Client C program file is compiled using gcc and has no command line parameters.

e.g :- ./client

Server C program file is compiled using gcc and has no command line parameters.

e.g :- ./server

When the client will connect with the server it will send the message to the server and server will return the reverse message back to the client.

The client simply asks user to input message and server after receiving it will return the reversed messaged.

**Program 2:**

Client C program file is compiled using gcc and has no command line parameters.

e.g :- ./client

Server C program file is compiled using gcc and has no command line parameters.

e.g :- ./server

When the client connects with the server it asks user to enter the parameters for the ls command. Then server will use that parameters with ls and sends the output of that command to client.

**Program 3:**

Client C program file is compiled using gcc and has no command line parameters.

e.g :- ./client

Server C program file is compiled using gcc and takes number of cylinders, number of sectors and name of file for disk as command line parameters.

e.g. :- ./server 4 2 disk.txt

When the client connects with the server it asks user to enter the valid commands to be executed on the basic file system. The server receives those commands and execute them on basic file system and send the particular output to the client user.

**Program 4:**

Client C program file is compiled using gcc and has no command line parameters.

e.g :- ./client

Server C program file is compiled using gcc and takes number of cylinders, number of sectors and name of file for disk as command line parameters.

e.g. :- ./server 4 2 disk.txt

When the client connects with the server it asks user to send the valid commands to be executed on the FAT system. The server receives those commands and execute them on FAT system and send the particular output to the client user.

**Program 5:**

Client C program file is compiled using gcc and has no command line parameters.

e.g :- ./client

Server C program file is compiled using gcc and takes number of cylinders, number of sectors and name of file for disk as command line parameters.

e.g. :- ./server 4 2 disk.txt

When the client connects with the server it asks user to send the valid commands to be executed on the FAT system. The server receives those commands and execute them on FAT system and send the particular output to the client user.

**Technical Manual:**

The disk is basically a file which is mapped using mmap(2) system call. When the user gives the file name then it is created if not already present and mapped using mmap call. It returns the pointer to the mapped file which is used for writing and reading data on it. So that file acts as a disk. In this way we are actually using the real storage space using file. So, whatever is written using the pointer returned by mmap that is written in actual file/ actual storage.

**Program 1:**

In Program 1 a function to reverse the message sent by the client is used in server. And that reversed message is sent back to the client.

**Program 2:**

The program 2 uses fork and exec family command to execute the ls with parameter passed to it by the client. Then the output of the ls is redirected using output redirection to the pipe. And from the pipe the output of ls is read and sent to the client.

**Program 3:**

In program 3, a struct is used to store the information about the disk such as disk size, number of cylinders, number of sectors, pointer to the disk. We used that struct to make the functions/ commands to work. In reading and writing we calculate the offset using the cylinder number and sector number in-order to write/read data from the exact block of the memory.

**Program 4:**

In program 4, a struct is used to store the information about the disk such as disk size, number of cylinders, number of sectors, pointer to the disk. A struct for file system is maintained in which Bitmap is used to keep track of the free blocks and number of files in the system. And a struct to maintain information about each file in the FAT system such as block number allocated to the particular file, its name, and size of file. The file system struct is used to make the function/commands work in sync with the disk. Whenever a block is allocated to the file the bit of that block in the Bitmap is set and when it is freed by the file, the block number is freed in the Bitmap and made available for writing. A bitmap is a char \* array so that each char can accommodate 8 blocks.

**Program 5:**

In program 5, a struct is used to store the information about the disk such as disk size, number of cylinders, number of sectors, pointer to the disk. A struct for file system is maintained in which Bitmap is used to keep track of the free blocks. And a struct to maintain information about each directory in the FAT system. A directory struct has info such as files in the directory, number of files in it, and name of directory. A struct for files is also present in directory struct to maintain the file name and the blocks allocated to the file and size of file. Whenever a block is allocated to the file the bit of that block in the Bitmap is set and when it is freed by the file, the block number is freed in the Bitmap and made available for writing. A bitmap is a char \* array so that each char can accommodate 8 blocks.