ReadMe

Contents of the file:

1. **Code Help**
2. **How to execute the program**

1. Code Help:

Different functions and system call used in our program:

- **read()**:

```
int  read(int handle, void *buffer, int nbyte);
```

The read() function attempts to read n bytes from the file associated with handle, and places the characters read into buffer.

The function returns the number of bytes read. On end-of-file, 0 is returned, on error it returns -1.

Header file needed for read():

```
#include <fcntl.h>
```

- **write()**:

```
int  write(int  handle, void  *buffer, int
nbyte);
```

The *write()* function attempts to write n bytes from *buffer* to the file associated with handle.

The function returns the number of bytes read. On end-of-file, 0 is returned, on error it returns -1.

Header file needed for write():

```
#include <fcntl.h>
```

❖ close():

```
int close(int handle);
```

The *close()* function closes the file associated with handle. The function returns 0 if successful, -1 to indicate an error.

Header file needed for close():

```
#include <fcntl.h>
```

❖ lseek():

Move the read/write file offset.

```
lseek(int fildes, off_t offset, int whence);
```

The *lseek*() function shall set the file offset for the open file description associated with the file descriptor *fildes,* as follows:

- If *whence* is SEEK_SET, the file offset shall be set to *offset* bytes.
- If *whence* is SEEK_CUR, the file offset shall be set to its current location plus *offset*.

Header file needed for lseek():

```
#include<unistd.h>
```

Upon successful completion, the resulting offset, as measured in bytes from the beginning of the file, shall be returned. Otherwise, *errno* shall be set to indicate the error, and the file offset shall remain unchanged.

❖ **`memset()`:**

```
void *memset(void *str, int c, size_t n)
```

It copies the character c (an unsigned char) to the first n characters of the string pointed to, by the argument str.

This function returns a pointer to the memory area str.

❖ **fflush():**

```
fflush(FILE *stream)
```

It flushes the output buffer of a stream.

stream − This is the pointer to a FILE object that specifies a buffered stream.

This function returns a zero value on success. If an error occurs, EOF is returned and the error indicator is set.

2. How to execute the program

We can format any storage device with our program including pen drive, hard disk to our file system.
Note: It is not advisable to format hard disk into our file system.

First of all we will convert our c code into executable program using the ***gcc*** command:

```
kashish@24-fd-52-96-4d-19:~/OS Project/OS_Project
File  Edit  View  Search  Terminal  Help
23:22 kashish@24-fd-52-96-4d-19 OS_Project $ ls
browse.c  format.c  header.h
23:22 kashish@24-fd-52-96-4d-19 OS_Project $ gcc format.c -o format
23:22 kashish@24-fd-52-96-4d-19 OS_Project $ gcc browse.c -o browse
23:22 kashish@24-fd-52-96-4d-19 OS_Project $ ls
browse  browse.c  format  format.c  header.h
23:22 kashish@24-fd-52-96-4d-19 OS_Project $
```

Formatting of pen drive is shown here.

For converting the pen drive into our file system, we need to know where our storage device is mounted on. That we can do by ***df*** command:

```
kashish@24-fd-52-96-4d-19:~/OS Project/OS_Project
File  Edit  View  Search  Terminal  Help
23:25 kashish@24-fd-52-96-4d-19 OS_Project $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          886700       0    886700   0% /dev
tmpfs             896640     260    896380   1% /dev/shm
tmpfs             896640    1332    895308   1% /run
tmpfs             896640       0    896640   0% /sys/fs/cgroup
/dev/sda1       81015560 9922440  66954728  13% /
tmpfs             896640     836    895804   1% /tmp
tmpfs             179332       8    179324   1% /run/user/42
tmpfs             179332      32    179300   1% /run/user/1000
/dev/sdb1        7815676   24252   7791424   1% /run/media/kashish/26F6DE8DF6DE5
C9D
23:25 kashish@24-fd-52-96-4d-19 OS_Project $
```

We can see that our pen drive named kashish is found at last. It is /dev/sdb1.

Now we have found our storage device. Then we will convert our pen drive in our file system.

But before that we need to be a root user of the system for accessing the file system of the storage device. That we can do by ***su -*** command which is as follows:

```
root@24-fd-52-96-4d-19:~
File  Edit  View  Search  Terminal  Help
23:27 kashish@24-fd-52-96-4d-19 OS_Project $ su -
Password:
23:27 root@24-fd-52-96-4d-19 ~ #
```

Now we will format the pen drive in our file system. For that we need to pass the /dev/sdb1 as the argument which we have found using ***df*** command.

```
root@24-fd-52-96-4d-19:/home/kashish/OS Project/OS_Project
File  Edit  View  Search  Terminal  Help
23:28 root@24-fd-52-96-4d-19 OS_Project # ./format /dev/sdb1
Partitioning 8003256320 byte sized /dev/sdb1 ... done
23:32 root@24-fd-52-96-4d-19 OS_Project #
```

This will convert our pen drive into our file system.

Now our pen drive is in our file system. Now we can explore/browse our file system.

```
root@24-fd-52-96-4d-19:/home/kashish/OS Project/OS_Project
File  Edit  View  Search  Terminal  Help
23:35 root@24-fd-52-96-4d-19 OS_Project # ./browse /dev/sdb1
Welcome to SFS Browsing Shell

Block size     : 512 bytes
Partition size : 15631360 blocks
File entry size: 64 bytes
Entry tbl size : 1563136 blocks
Entry count    : 12505088

 $>
```

- We can enter **?** Command to get the supported commands.

```
root@24-fd-52-96-4d-19:/home/kashish/OS Project/OS_Project
File  Edit  View  Search  Terminal  Help
23:35 root@24-fd-52-96-4d-19 OS_Project # ./browse /dev/sdb1
Welcome to SFS Browsing Shell

Block size     : 512 bytes
Partition size : 15631360 blocks
File entry size: 64 bytes
Entry tbl size : 1563136 blocks
Entry count    : 12505088

 $> ?
Supported commands:
        ?       quit    list    create <file>   remove <file>
                chperm <0-7> <file>     read <file>     write <file>
 $>
```

- We can create any file using the ***create*** command and using the ***list*** command we can list all the files. Format of create command is: **create <file_name>**

```
 $> ?
Supported commands:
        ?       quit    list    create <file>   remove <file>
                chperm <0-7> <file>     read <file>     write <file>
 $> create abc
 $> list
abc                     0 bytes  rwx  Wed Jul 20 20:58:28 2968
 $>
```

- We can write into any of the present file using the write. The format of the write command is:
  write <file_name>
  file_name: name of the in which you want to write.
  We can press ***ctrl+d*** to finish the writing.

```
 $> list
abc                     0 bytes  rwx  Wed Jul 20 20:58:28 2968
xyz                     0 bytes  rwx  Wed Jul 20 21:01:25 2968
 $> write abc
Life begins at the end of your comfort zone.
 $>
```

- We can read from any file through the ***read*** command. The format of the read command is:
  read <file_name>
  file_name: name of the file from which you want to read.

```
 $> read abc
Life begins at the end of your comfort zone.
 $>
```

- We can change permission of any file using ***chperm*** command. The format of the chperm command is:
  chperm <0-7> <file_name>
  file_name: name of the file of which we want to change the permission
  0 – No permission
  1 – Only execute permission
  2 – Only Write permission
  3 – Write and Execute permission
  4 – Only Read permission
  5 – Read and Execute permission
  6 – Read and Write permission
  7 – Read, Write and Execute permission
  Note: Default permission of the file is Read, Write and Execute (7).

```
 $> chperm 5 xyz
 $> list
abc                     45 bytes  rwx  Wed Jul 20 21:04:18 2968
xyz                      0 bytes  r-x  Thu May  7 08:04:53 2696
 $>
```

You can see that permission of xyz file has been changed to Read and Execute.

- We can remove any file from all the available files using ***remove*** command. The format of the remove command is:
  remove <file_name>
  file_name: name of the file we want to remove.

```
 $> remove xyz
 $> list
abc                     45 bytes  rwx  Wed Jul 20 21:04:18 2968
 $>
```

It can be seen that file named **xyz** has been removed.

- If we enter any wrong or misspelled command then it will show the proper error message and prints all the supported commands.

```
Unknown/Incorrect command: hello
Supported commands:
        ?       quit    list    create <file>   remove <file>
                chperm <0-7> <file>     read <file>     write <file>
 $> crete
Unknown/Incorrect command: crete
Supported commands:
        ?       quit    list    create <file>   remove <file>
                chperm <0-7> <file>     read <file>     write <file>
 $>
```

- Now if user want to quit from our interface, he/she can simply enter ***quit*** command and he/she will be out of the interface.

```
 $> quit
00:08 root@24-fd-52-96-4d-19 OS_Project #
```

That's how our file system works.

Thanks.