

# OS Project1 Example

521030910229 Hao Chiyu

May 25, 2023

## Contents

|          |                               |           |
|----------|-------------------------------|-----------|
| <b>1</b> | <b>Step 1</b>                 | <b>2</b>  |
| 1.1      | Normal output . . . . .       | 2         |
| 1.2      | Error throwing . . . . .      | 3         |
| <b>2</b> | <b>Step 2</b>                 | <b>5</b>  |
| 2.1      | f & mk & mkdir & ls . . . . . | 5         |
| 2.2      | cd . . . . .                  | 5         |
| 2.3      | rm & rmdir . . . . .          | 6         |
| 2.4      | cat & w & i & d & e . . . . . | 8         |
| 2.5      | Error throwing . . . . .      | 9         |
| <b>3</b> | <b>Step 3</b>                 | <b>12</b> |
| 3.1      | Setup . . . . .               | 12        |
| 3.2      | f & mk & mkdir & ls . . . . . | 13        |
| 3.3      | cd . . . . .                  | 13        |
| 3.4      | rm & rmdir . . . . .          | 13        |
| 3.5      | cat & w & i & d & e . . . . . | 14        |
| 3.6      | Client reconnect . . . . .    | 15        |
| 3.7      | Persistence . . . . .         | 16        |
| 3.8      | Error throwing . . . . .      | 16        |

# 1 Step 1

I think reading before writing is undefined behavior, so the result of program may be different from demo: "R 2 3" before "W 2 3" will print "NO". :)

## 1.1 Normal output

As can be seen in Figure 1 & 2, the program can receive argument and build a disk system we want. It can support "I", "R" and "W" operation, then print messages into stdout and log.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 c ./disk 4 8 1 disk_storage
I
W 1 2 Hello. world!
R 1 2
YES: Hello. world!
E
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c
```

Figure 1: First run of disk program.

```
1 4 8
2 YES
3 YES: Hello. world!
4 Goodbye!
```

Figure 2: Log of first run.

After shutdown program, we can restart the program and use "disk\_storage" as the storage file again. In Figure 3 & 4, we can see program read and modified the content we stored in the first run.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 8 1 disk_storage
R 1 2
YES: Hello, world!
W 1 2 test test
R 1 2
YES: test test
E
```

Figure 3: First run of disk program.

```
1 YES: Hello, world!
2 YES
3 YES: test test
4 Goodbye!
```

Figure 4: Log of first run.

## 1.2 Error throwing

From Figure 5, we can find that the program printed error message and exited gracefully when the number or format of arguments is incorrect.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 8 1 disk_storage wrong_arg
Usage: ./disk <#cylinders> <#sector per cylinder> <track-to-track delay> <#disk-storage-filename>

~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 0 1 disk_storage
Only positive number args are supported.

~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 aaa 1 disk_storage
Only positive number args are supported.
```

Figure 5: Error msg when arguments of program are incorrect.

The program also sends error messages when arguments of commands are invalid, which can be seen in Figure 6 & 7.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c rm disk_storage

~/Desktop/courses/os/lab/project3/step1 c master !4 c ./disk 4 8 1 disk_storage
I 1
Error: invalid request.
W 1 2
Error: invalid request.
W -1 2 test
Error: invalid request.
W 1 2 test
R 1
Error: invalid request.
R 1 2 3
Error: invalid request.
E
```

Figure 6: Error msg when arguments of command are incorrect.

```
1 NO
2 NO
3 NO
4 YES
5 NO
6 NO
7 Goodbye!
```

Figure 7: Log when arguments of command are incorrect.

Read from & write to sectors which do not exist will cause error.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 8 1 disk_storage
W 5 6 hello
Error: invalid request.
R 5 6
Error: invalid request.
```

Figure 8: Error: R/W of sector which don't exist.

At last, we can see in Figure 9 that, the program will reject accesses when the sector is invalid, which means that this sector has not been written.

```
~/Desktop/courses/os/lab/project3/step1 c master !4 ?1 c ./disk 4 8 1 disk_storage
R 0 2
Error: invalid request.
E
```

Figure 9: Error msg when try to access invalid sector.

## 2 Step 2

### 2.1 f & mk & mkdir & ls

In Figure 9 and 10, we use "f" command to initialize the file system, then we create 3 files and 2 directories with "mk" and "mkdir" commands. At last, "ls" helps us view files and dirs in the current directory.

```
- Desktop/courses/os/lab/project3/step2 c master !3 ?3 c ./fs
> f
> mk a
> mk b
> mk c
> mkdir hello
> mkdir world
> ls
file a 0 Last Update Time: 2023/5/25 21:51
file b 0 Last Update Time: 2023/5/25 21:51
file c 0 Last Update Time: 2023/5/25 21:51
dir hello 256 Last Update Time: 2023/5/25 21:51
dir world 256 Last Update Time: 2023/5/25 21:51
```

Figure 10: Outputs of f, mk, mkdir and ls commands.

```
YES: file system initied
YES: create file a
YES: create file b
YES: create file c
YES: create directory hello
YES: create directory world
file a 0 Last Update Time: 2023/5/25 21:51
file b 0 Last Update Time: 2023/5/25 21:51
file c 0 Last Update Time: 2023/5/25 21:51
dir hello 256 Last Update Time: 2023/5/25 21:51
dir world 256 Last Update Time: 2023/5/25 21:51
```

Figure 11: Logs of above operations.

### 2.2 cd

Figure 11 & 12 show the output of command "cd". This time, we will respectively enter the "hello" and "world" dirs and create "cat" and "dog" files in these two subdirectories. The first and third "cd" commands use relative paths, while the rest use absolute paths. At the same time, we can also find that programs can handle "." and "..".

```

> cd hello
> mk cat
> cd /world
> mk dog
> cd cd ../../hello
Error: invalid argument.
> cd ../../hello
> ls
file  cat    0    Last Update Time: 2023/5/25 21:51
> cd ../hello/../../world
> ls
file  dog    0    Last Update Time: 2023/5/25 21:51

```

Figure 12: Usage of cd.

```

YES
YES: create file cat
YES
YES: create file dog
YES
file  cat    0    Last Update Time: 2023/5/25 21:51
YES
file  dog    0    Last Update Time: 2023/5/25 21:51

```

Figure 13: Logs of above operations.

## 2.3 rm & rmdir

In figure 13 & 14, we use "rm" and "rmdir" commands delete file "a" and dir "hello".

```
> cd /
> ls
file a    0    Last Update Time: 2023/5/25 21:51
file b    0    Last Update Time: 2023/5/25 21:51
file c    0    Last Update Time: 2023/5/25 21:51
dir hello 256   Last Update Time: 2023/5/25 21:51
dir world 256   Last Update Time: 2023/5/25 21:51
> rm a
> rmdir hello
> ls
file b    0    Last Update Time: 2023/5/25 21:51
file c    0    Last Update Time: 2023/5/25 21:51
dir world 256   Last Update Time: 2023/5/25 21:51
```

Figure 14: Usage of rm and rmdir.

```
YES
file a    0    Last Update Time: 2023/5/25 21:51
file b    0    Last Update Time: 2023/5/25 21:51
file c    0    Last Update Time: 2023/5/25 21:51
dir hello 256   Last Update Time: 2023/5/25 21:51
dir world 256   Last Update Time: 2023/5/25 21:51
YES: remove file successfully.
YES: remove directory successfully.
file b    0    Last Update Time: 2023/5/25 21:51
file c    0    Last Update Time: 2023/5/25 21:51
dir world 256   Last Update Time: 2023/5/25 21:51
```

Figure 15: Logs of above operations.

## 2.4 cat & w & i & d & e

In Figure 15 and 16, we use command "i", "w" and "d" modify the content of file, and use "cat" view the content. At last, we use "e" to shutdown the program.

```
> w b 100 hello, world!
> cat b
hello, world!
> d b 2 4
> cat b
he world!
> i b 2 9 something
> cat b
hesomething world!
> i b 200 7 append
> cat b
hesomething world!append
> d b 1 1000
> cat b
h
> e

~/Desktop/courses/os/lab/project3/step2 c master !4 ?3 c
```

Figure 16: Usage of cat & w & i & d & e.

```
YES: write to file successfully.
YES:
hello, world!
YES: delete file content successfully!
YES:
he world!
YES: insert to file successfully.
YES:
hesomething world!
YES: insert to file successfully.
YES:
hesomething world!append
YES: delete file content successfully!
YES:
h
Goodbye!
```

Figure 17: Logs of above operations.



## 2.5 Error throwing

From Figure 17, we can find that the program printed error message and exited gracefully when the number or format of arguments is incorrect.

```
~/Desktop/courses/os/lab/project3/step2 c master !3 ?2 c ./fs 1
Usage: ./fs
```

Figure 18: Error msg when arguments of program are incorrect.

The program will print error msg when you try running other commands before initialize file system, as can be seen in Figure 18 & 19.

```
~/Desktop/courses/os/lab/project3/step2 c master !3 ?2 c ./fs
> mk a
Error: file system not initialized.
```

Figure 19: Error when run command before system initialized.

```
1 NO: file system not initialized.
```

Figure 20: Logs of above error.

The program will print error msg when you try creating file/dir which name has been occupied as can be seen in Figure 20 & 21.

```
> f
> mk a
> mk a
Error: file name already exists.
> mkdir b
> mkdir b
Error: dir name already exists.
```

Figure 21: Error when create file with already existing name.

```
2 YES: file system initied
3 YES: create file a
4 NO: file name already exists.
5 YES: create directory b
6 NO: dir name already exists.
```

Figure 22: Logs of above error.

The program will print error msg when your command has wrong arguments, as can be seen in Figure 22 & 23.

```
> mk
Error: invalid argument.
> mkdir
Error: invalid argument.
> rm
Error: invalid argument.
> rmdir
Error: invalid argument.
> w a
Error: invalid argument.
> i a
Error: invalid argument.
> d a
Error: invalid argument.
> cat
Error: invalid argument.
> ls aaa
Error: invalid argument.
```

Figure 23: Error when commands have wrong number of argument.

```
7 NO: invalid argument
8 NO: invalid argument
9 NO: invalid argument
10 NO: invalid argument
11 NO: invalid argument
12 NO: invalid argument
13 NO: invalid argument
14 NO: invalid argument
15 NO: invalid argument
```

Figure 24: Logs of above error.

The program will print error msg when your command can't find file or dir you want to access, as can be seen in Figure 24 & 25.

```
> rm c
Error: file not found.
> rmdir c
Error: directory not found.
> i c 1 1 a
Error: file not found.
> w c 1 1
Error: file not found.
> d c 1 1
Error: file not found.
> cat c
Error: file not found.
> cd c
Error: not found.
```

Figure 25: Error when access file/dir not existing.

```
16 NO: file not found.
17 NO: directory not found.
18 NO: file not found.
19 NO: file not found.
20 NO: file not found.
21 NO: file not found.
22 NO: not found.
```

Figure 26: Logs of above error.

The program will print error msg when offset/length arguments are invalid, as can be seen in Figure 26 & 27.

```
~/Desktop/courses/os/lab/project3/step2 c master !2 ?2 c ./fs
> f
> mk a
> i a -1 test
Error: invalid offset.
> w a -1 test
Error: invalid argument.
> d a -1 -1
Error: invalid argument.
_
```

Figure 27: Error when command arguments are invalid.

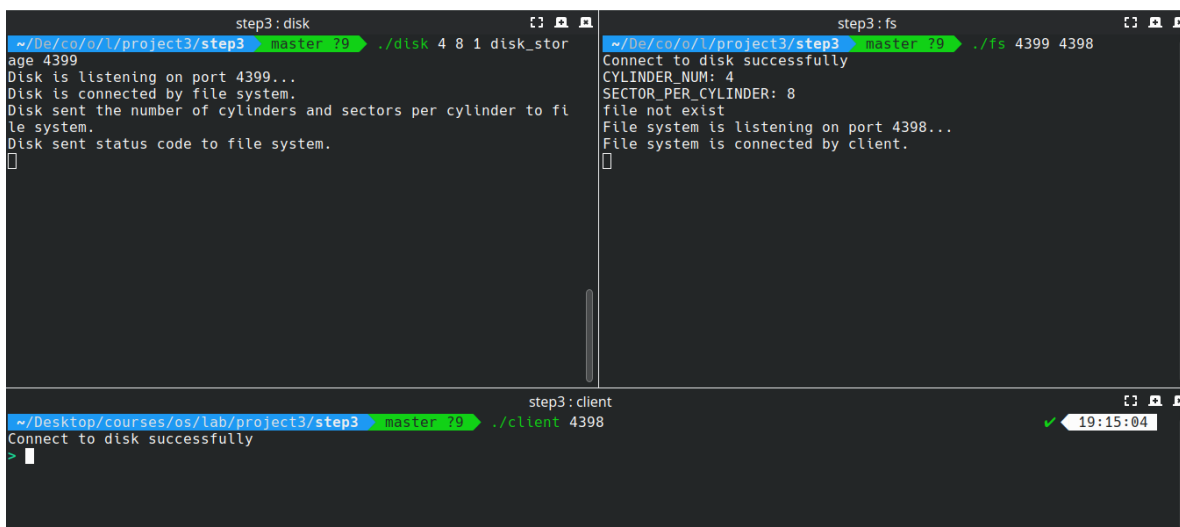
```
23 NO: invalid offset
24 NO: invalid argument
25 NO: invalid argument
26 Goodbye!
```

Figure 28: Logs of above error.

## 3 Step 3

### 3.1 Setup

Run disk program first, then set up file system program and connects to disk. At last, run client program.



```
step3: disk
~/Desktop/courses/os/lab/project3/step3 master ?9 ./disk 4 8 1 disk_stor
age 4399
Disk is listening on port 4399...
Disk is connected by file system.
Disk sent the number of cylinders and sectors per cylinder to fl
le system.
Disk sent status code to file system.
[]

step3: fs
~/Desktop/courses/os/lab/project3/step3 master ?9 ./fs 4399 4398
Connect to disk successfully
CYLINDER_NUM: 4
SECTOR_PER_CYLINDER: 8
file not exist
File system is listening on port 4398...
File system is connected by client.
[]

step3: client
~/Desktop/courses/os/lab/project3/step3 master ?9 ./client 4398
Connect to disk successfully
> [✓ 19:15:04]
```

Figure 29: Set up all programs.

### 3.2 f & mk & mkdir & ls

In Figure 29, we use "f" command to initialize the file system, then we create 3 files and 2 directories with "mk" and "mkdir" commands. At last, "ls" helps us view files and dirs in the current directory.

```
> f
Success: file system initialized.
> mk a
> mk b
> mk c
> mkdir hello
> mkdir world
> ls
file  a      0      Last Update Time: 2023/5/25 21:58
file  b      0      Last Update Time: 2023/5/25 21:58
file  c      0      Last Update Time: 2023/5/25 21:58
dir   hello  256     Last Update Time: 2023/5/25 21:58
dir   world  256     Last Update Time: 2023/5/25 21:58
```

Figure 30: Outputs of f, mk, mkdir and ls commands.

### 3.3 cd

Figure 30 show the output of command "cd". This time, we will respectively enter the "hello" and "world" dirs and create "cat" and "dog" files in these two subdirectories. The first and third "cd" commands use relative paths, while the rest use absolute paths. At the same time, we can also find that programs can handle "." and "..".

```
> cd hello
> mk cat
> cd /world
> mk dog
> cd ../../hello
> ls
file  cat      0      Last Update Time: 2023/5/25 21:58
> cd ../hello/../../world
> ls
file  dog      0      Last Update Time: 2023/5/25 21:58
```

Figure 31: Usage of cd.

### 3.4 rm & rmdir

In figure 31, we use "rm" and "rmdir" commands delete file "a" and dir "hello".

```

> cd /
> ls
file a 0 Last Update Time: 2023/5/25 21:58
file b 0 Last Update Time: 2023/5/25 21:58
file c 0 Last Update Time: 2023/5/25 21:58
dir hello 256 Last Update Time: 2023/5/25 21:58
dir world 256 Last Update Time: 2023/5/25 21:58
> rm a
> rmdir hello
> ls
file b 0 Last Update Time: 2023/5/25 21:58
file c 0 Last Update Time: 2023/5/25 21:58
dir world 256 Last Update Time: 2023/5/25 21:58

```

Figure 32: Usage of rm and rmdir.

### 3.5 cat & w & i & d & e

In Figure 32, we use command "i", "w" and "d" modify the content of file, and use "cat" view the content. At last, we use "e" to shutdown the program.

```

> w b 100 hello, world!
> cat b
hello, world!
> d b 2 4
> cat b
he world!
> i b 2 9 something
> cat b
hesomething world!
> i b 100 7 append
> cat b
hesomething world!append
> d b 1 100
> cat b
h
> e
Exit

```

Figure 33: Usage of cat & w & i & d & e.

### 3.6 Client reconnect

In Figure 33, we exit client program and restart it, this time the file system program listens to port 4401. After connection established, we write some words into file "b", which is created in previous run, then we exit again. At last connection, we read "b" and the program print the content we write before.

```
~/De/co/o/l/project3/step3 fix !1 ?9 ./client 4401
Connect to disk successfully
> i b 0 persistence test
> e
Exit

~/De/co/o/l/project3/step3 fix !1 ?9 ./client 4402
Connect to disk successfully
> cat b
persistence test
> e
Exit
```

Figure 34: Reconnect: client view point.

```
File system is listening on port 4401...
File system is connected by client.
Command: i b 0 persistence test
Command: exit
Client exit.
Retry port 4402
File system is listening on port 4402...
File system is connected by client.
Command: cat b
persistence test
Command: exit
Client exit.
Retry port 4403
File system is listening on port 4403...
```

Figure 35: Reconnect: file system view point.

### 3.7 Persistence

After 3.6 subsection, we reconnect and type "q" in file system, it will exit after client exits. Then we restart all 3 programs and read b, the program will print content we wrote before.

```
~/De/co/o/lab/project3/step3 fix !1 ?9 ./client 4403
Connect to disk successfully
> cat b
persistence test
> □
```

Figure 36: Persistence test: client view point.

```
File system is listening on port 4403...
File system is connected by client.
q
Command: exit
Waiting for client exiting...

~/De/co/o/l/project3/step3 fix !1 ?9 ./fs 4399 4403
Connect to disk successfully
CYLINDER_NUM: 4
SECTOR_PER_CYLINDER: 8
file exist
File system is listening on port 4403...
File system is connected by client.
Command: cat b
□
```

Figure 37: Persistence: file system view point.

### 3.8 Error throwing

The program will check the number and validity of arguments.

```
~/De/co/o/lab/project3/step3 fix !1 ?9 ./client 4403 1
Usage: ./client <FSPort>
```

Figure 38: Error: invalid program argument.

File system/client will shutdown if they cannot connect to disk/files system in given time duration.

```
~/De/co/o/l/project3/step3 fix !1 ?9 ./fs 4380 4398
Retry...
Retry...
Retry...
Retry...
Connect failed, please set up disk.
```

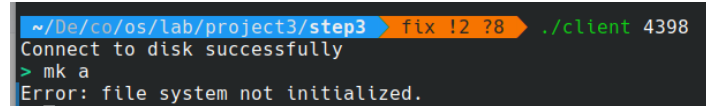
Figure 39: Error: file system cannot connect to disk.

```
~/De/co/os/lab/project3/step3 fix !2 ?9 ./client 4403
Retry...
Retry...
Retry...
Retry...
Connect failed, please set up file system.
```

Figure 40: Error: client cannot connect to file system.



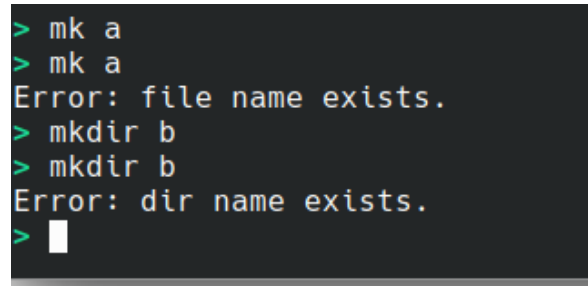
The program will print error msg when you try running other commands before initialize file system.

A terminal window with a dark background. The prompt is '~ /De/co/os/lab/project3/step3' followed by a blue arrow pointing right with the text 'fix !2 ?8'. To the right of the arrow is the text './client 4398'. Below the prompt, the text 'Connect to disk successfully' is displayed. Then, the command '> mk a' is entered, and the response 'Error: file system not initialized.' is shown.

```
~/De/co/os/lab/project3/step3 fix !2 ?8 ./client 4398
Connect to disk successfully
> mk a
Error: file system not initialized.
```

Figure 41: Error when run command before system initialized.

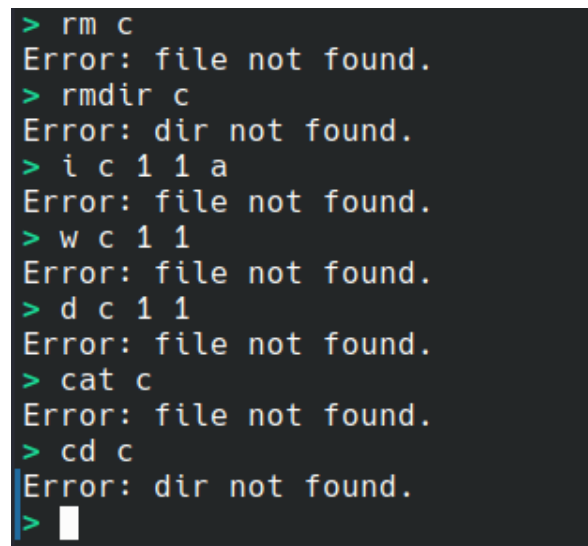
The program will print error msg when you try creating file/dir which name has been occupied.

A terminal window with a dark background. The prompt is '>'. The command 'mk a' is entered, followed by 'mk a' again. The response is 'Error: file name exists.'. Then, 'mkdir b' is entered, followed by 'mkdir b' again. The response is 'Error: dir name exists.'. The prompt '>' is shown at the end.

```
> mk a
> mk a
Error: file name exists.
> mkdir b
> mkdir b
Error: dir name exists.
>
```

Figure 42: Error when create file with already existing name.

The program will print error msg when your command has wrong arguments.

A terminal window with a dark background. The prompt is '>'. The command 'rm c' is entered, followed by 'rmdir c'. Both result in 'Error: file not found.' and 'Error: dir not found.' respectively. Then, 'i c 1 1 a' is entered, followed by 'w c 1 1', 'd c 1 1', 'cat c', and 'cd c'. All these result in 'Error: file not found.' and 'Error: dir not found.' respectively. The prompt '>' is shown at the end.

```
> rm c
Error: file not found.
> rmdir c
Error: dir not found.
> i c 1 1 a
Error: file not found.
> w c 1 1
Error: file not found.
> d c 1 1
Error: file not found.
> cat c
Error: file not found.
> cd c
Error: dir not found.
>
```

Figure 43: Error when commands have wrong number of argument.

The program will print error msg when your command can't find file or dir you want to access.

```
> rm c
Error: file not found.
> rmdir c
Error: dir not found.
> i c 1 1
Error: file not found.
> w c 1 1
Error: file not found.
> d c 1 1
Error: file not found.
> cat c
Error: file not found.
> cd c
Error: dir not found.
> 
```

Figure 44: Error when access file/dir not existing.

The program will print error msg when offset/length arguments are invalid.

```
~/De/co/os/lab/project3/step3 > fix !1 ?8 > ./client 4398
Connect to disk successfully
> f
Success: file system initialized.
> mk a
> i a -1 test
Error: this argument must be a non-negative integer.
> w a -1 test
Error: this argument must be a non-negative integer.
> d a -1 2
Error: this argument must be a non-negative integer.
> 
```

Figure 45: Error when command arguments are invalid.