Write up of ASG0: mycat

Yujia Li
CruzID: yli302

How does the code for handling a file differ from that for handling standard input? What concept is this an example of?

For handling a file, the program needs to `open()` the file depends on file and `read()` from **file descriptor.**

For handling a standard input, the program needs to `read()` from the **standard input.**

Because of the different way to handle user data, I set them in this structure:

```
if ()  {standard input} else {file};
```

This is an example of **modularity.**

The way to write the read result of file descriptor and stdin in `mycat` code is same, I just need to change the first argument in `read()` function.

The input to `mycat` from file and standard input could be different, but the outputs are always in the same format. This stands '*Be tolerant of inputs and strict on outputs'.*

Therefore, it is a example of **abstraction**.

TEST:

1. With zero argument:

   Copy stdin to stdout.

   ```
   $ ./mycat
   first input
   first input
   second input
   second input

   $ ./mycat < smallfile.txt
   This is a small file.$

   $ ./mycat < test
   cat: -: Is a directory
   ```

2. With one argument:
   a. File less than 10KB

      ```
      $ ./mycat smallfile.txt
      This is a small file.$
      ```

   b. File greater than 1MB

      Just like smallfile, but more things on standard output.

c. File of size zero

   End program without output.

   ```
   $ ./mycat zerofile.txt
   $
   ```

d. File name which does not exist.

   ```
   $ ./mycat fff
   cat: fff: No such file or directory
   ```

e. File name which is a directory

   ```
   $ ./mycat test
   cat: test: Is a directory
   ```

f. File which do not have read permission.

   It should be `Permission denied`

g. Other test with redirection operator

   ```
   $ ./mycat smallfile.txt >> newfile
   $ ./mycat newfile
   This is a small file.$ ./mycat smallfile.txt >>
   newfile
   This is a small file.This is a small file.$
   ```

3. With more than one arguments:

   Program will run the first argument, then run the second argument, and until all arguments have run.