Bo Liu

How to test

Use test.sh
cat filename > out.cat
./dog filename >out.dog
Differ out.cat out.dog

Running test.sh with replacement of filename, it will report differ if there are difference between outputs generated by two function such as

cat test1.txt > out.cat
./dog test1.txt >out.dog
Differ out.cat out.dog

Run test.sh on the command line, it will tell if there are differences between outputs generated by two functions.

By creating some random text file or binary file and replacing the filename in the test.sh, and running the test.sh on the command line, this test is whole-system testing.

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, we first call open() : int open(const char * pathname, int flags), it will return the file descriptor. Then we call the read(): read(int fd, void *buf, size_t count), it will store the content from the file descriptor into the buffer.

For handling standard input, we do not need to call open () to open a file, since we get input from the user such as read(0,buffer,size of (buffer), number 0 here as indicator tell the read(), we get input from the user and store it into buffer.

Concept：from manul page:
read() attempts to read up to *count* bytes from file descriptor *fd* into the buffer starting at buf. This shows that since files and the keyboard are not the same things, the read() function generalizes these two things into just sequences of bytes. From many other aspects, if we write code "program" to any sequence of bytes, we can reuse that generalized code in many places. This fact reveals that for different objects, we can convert it into sequences of byte which is a kind of generalized way. Abstraction is the concept how the computer handle these different object