Preparation for Lab 4

Read through `man 2 open`.

- What header file do I need to include? How do I know? How can I look this up for other functions using `man`?

If I open a file with the following call to open: `open("fileexists.txt", O_RDONLY)`

- What is the return type of `open`? i.e., what is a file descriptor to your program?
- What are the read/write limitations of that file descriptor?

What if I call `open("this_file_doesnt_exist.txt", O_RDONLY)`?

- What is the returned value?
- Why did it return that?

What if I call `open("make_a_file.txt", O_RDONLY | O_CREAT)`?

- Why can't I write to it?

Okay, I change the above call to `open("make_a_file.txt", O_RDWR | O_CREAT)` (assuming "make_a_file.txt doesn't exist yet").

- Why can't I write to it? I passed the read/write flag!!!
- Try issuing `$ ls -l` from the commandline and look at the permissions. does that help explain?

Okay, one more change! `open("make_a_file.txt", O_RDWR | O_CREAT, S_IRUSR | S_IWUSR)`

- What are those things?
- Why can I suddenly write to it?

I have a file "somefile.txt", and the string "hello world!" is saved in "somefile.txt" - What does the file contain if I call `open("somefile.txt", O_RDWR)`? - What if I call `open("somefile.txt", O_RDWR | O_TRUNC)`? open in your text editor and check

---

Read through `man 2 read`, `man 2 write`, `man 2 lseek`.

For the remainder of this section, lets assume that I start with a file "fileexists.txt" that has the string "the meaning of life is 42" inside.

I have a file descriptor as such:

```
int fd = open("fileexists.txt", O_RDWR);
```

I also have a buffer of length 128 (long enough to fit what we need) as follows:

```
char buf[128];

memset(buf, '\0', 128);
```

So the buffer `buf` contains all zeros.

If I run the following code snippet:

```
read(fd, buf, 4);

printf("'%s'\n", buf);
```

- What happens? what is printed out?

If I then run the following code snippet:

```
read(fd, buf, 8);

printf("'%s'\n", buf);
```

- What happens? What is printed out?
- Why is it printing what it is?

I then run the following code block:

```
write(fd, "of dogs", 7);
```

- Open the file in your text editor.
- What's in the file? why?
- Why oh why would it write something there?

Fix the file so it reads as it did before, containing "the meaning of life is 42"

Now, intead of the previous block of code, what if we instead ran the following block:

```
lseek(fd, 0, SEEK_SET);

write(fd, "a  ", 3);
```

- What's the difference from before? Why did it do what it did?

This tells us that a file descriptor points to some thing in the operating system that keeps track of the current offset within a file.

Read through `man 2 close`

Any time you create a file descriptor with `int fd = open(...)`, make sure you have a corresponding call to `close(fd)`, similar to the pairing of `malloc` and `free`.

How can I tell if a file exists? How can I get information on a file?

Read through `man 2 stat`

- What header file do I need to include? How do I know? How can I look this up for other functions using `man`?

Suppose I have a variable:

```
struct stat statbuf;
```

I run the following code snippet:

```
stat("file_exists.txt", &statbuf);
```

- What is returned from the call to stat?

What if I instead call the following code snippet?

```
stat("file_doesnt_exist.txt", &statbuf);
```

- What gets returned from the call to stat?
- How can I use this information to tell if a file exists?
- Why do I declare statbuf as such, but then reference it with `&`? why not just make a `struct stat *` instead?
- According to the man pages, how can I access the size of the file using statbuf? How about the permissions of the file?

Look through `man 2 stat` again carefully.

- How can I get the statistics (stat) on a file if I already have the file descriptor?

For all of the above, try to break it. Try `lseek` beyond the end of a file. Before the beginning (negative values)? How about opening twice, where the second call truncates it? how about if we do the following:

```
int fd_a, fd_b;

fd_a = open("somefile.txt", O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);

write(fd_a, "somestring", strlen("somestring"));

fd_b = open("somefile.txt", O_RDWR | O_TRUNC);

/* I intended to leave fd_a here */
write(fd_a, "someotherstring", strlen("someotherstring"));

close(fd_a);

close(fd_b);
```

What is in the file? Does it even run? What did you expect? What happened?

Some other questions:

- What is the return value of read/write if I don't have the proper permissions on the file? What if it fails? How can I check?
- Can I assume that a call to `open`/`close`/`read`/`write`/`stat`/`lseek`/etc will work? How will my grade look if I forget to check for error conditions on a function, and the TAs throw an edge case at my code?
- Can I ask the TAs about edge case handling? Can I ask the TAs about what edge cases to look out for? (hint: the answer is yes! please do!)

Tip:

If you're not sure where in the man pages to look, there's a handy command called `apropos` that lets you search the man pages for keywords. Try it on `getopt`:

```
$ apropos getopt
```

It will list out everywhere that `getopt` is mentioned, so you can narrow down your search. You will never need to google code for this class again!