Natalie Petrosian

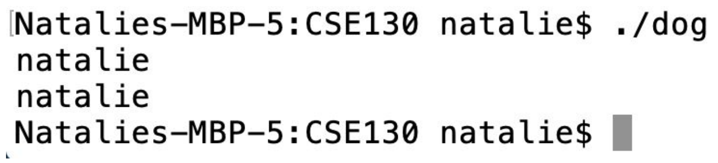
CSE 130 Principles of Computer System Design

Assignment 0

11 April 2020

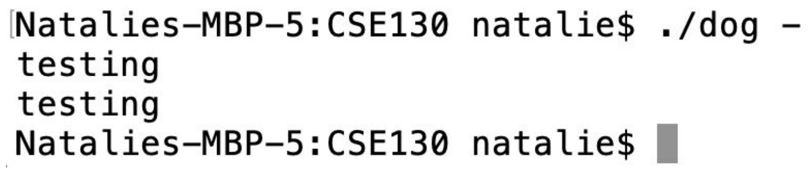
Testing:

<u>In this example, I tested ‘dog’ with no parameters:</u>



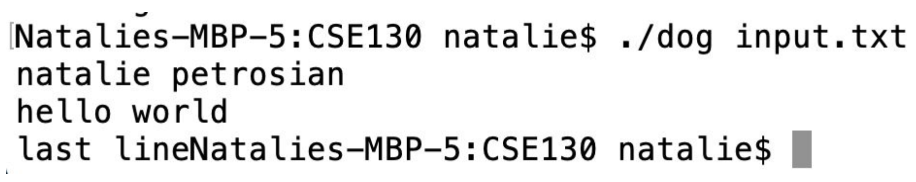
```
Natalies-MBP-5:CSE130 natalie$ ./dog
natalie
natalie
Natalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with the parameter - (dash):</u>



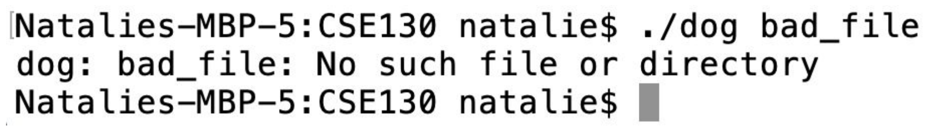
```
Natalies-MBP-5:CSE130 natalie$ ./dog -
testing
testing
Natalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with an existing file:</u>



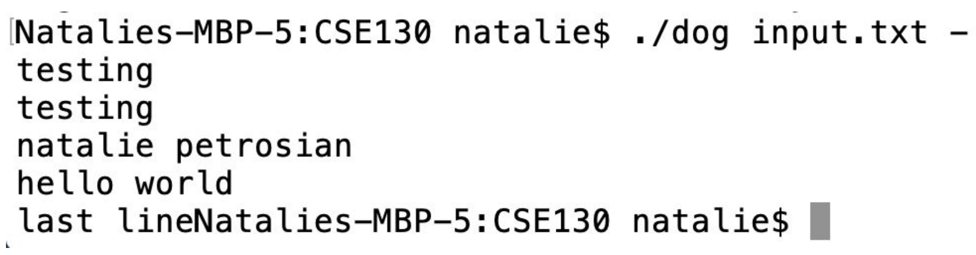
```
Natalies-MBP-5:CSE130 natalie$ ./dog input.txt
natalie petrosian
hello world
last lineNatalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with a non-existent file:</u>



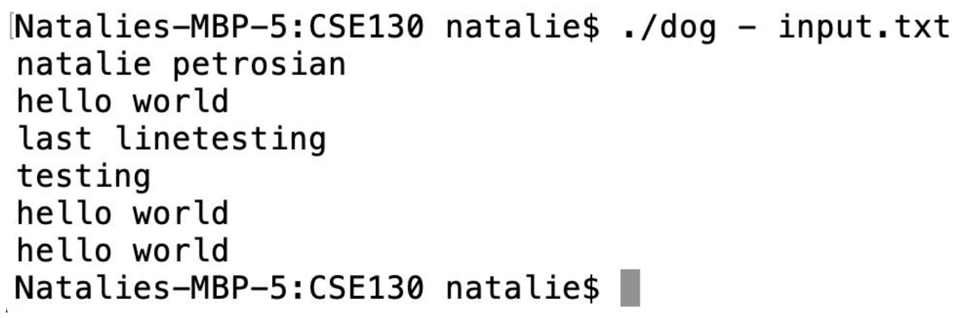
```
Natalies-MBP-5:CSE130 natalie$ ./dog bad_file
dog: bad_file: No such file or directory
Natalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with an existing file and a - (dash):</u>



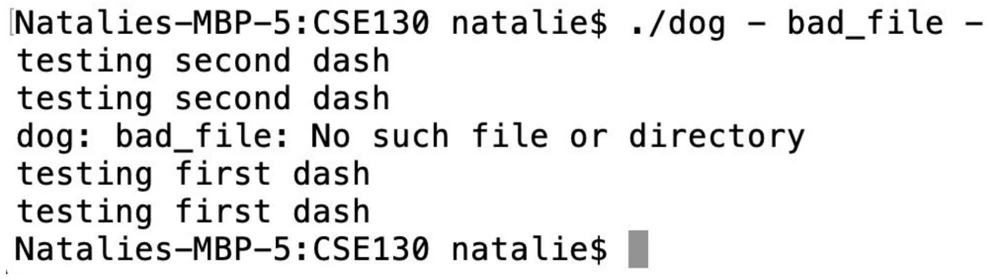
```
Natalies-MBP-5:CSE130 natalie$ ./dog input.txt -
testing
testing
natalie petrosian
hello world
last lineNatalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with a - (dash) and an existing file:</u>



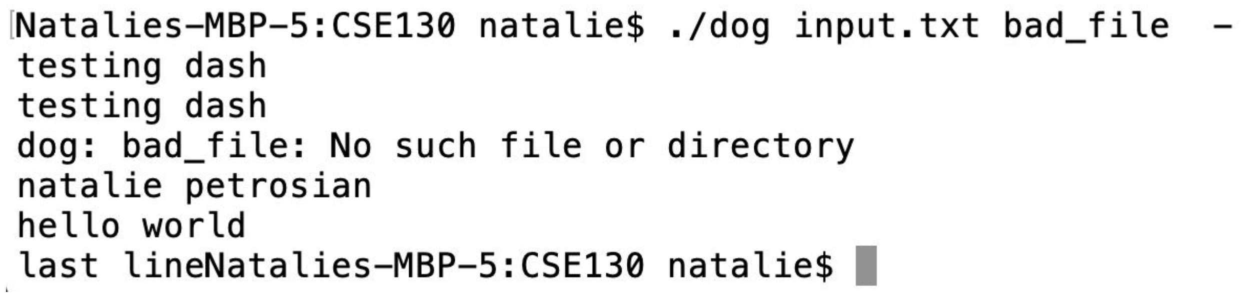
```
Natalies-MBP-5:CSE130 natalie$ ./dog - input.txt
natalie petrosian
hello world
last linetesting
testing
hello world
hello world
Natalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with two dashes and a bad file:</u>



```
Natalies-MBP-5:CSE130 natalie$ ./dog - bad_file -
testing second dash
testing second dash
dog: bad_file: No such file or directory
testing first dash
testing first dash
Natalies-MBP-5:CSE130 natalie$ 
```

<u>In this example, I tested ‘dog’ with an existing file, a non-existent file, and a - (dash):</u>



```
Natalies-MBP-5:CSE130 natalie$ ./dog input.txt bad_file  -
testing dash
testing dash
dog: bad_file: No such file or directory
natalie petrosian
hello world
last lineNatalies-MBP-5:CSE130 natalie$ 
```

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

Answer: In this assignment, all input and output are handled by file descriptors. In a Unix system, a file descriptor is a non-negative integer. There are three file descriptors that are reserved as follows:

Integer value 0: Standard input defined as STDIN_FILENO
Integer value 1: Standard output defined as STDOUT_FILENO
Integer value 2: Standard error defined as STDERR_FILENO

When the open() library call is used to open an actual file, the next available integer is assigned to that file descriptor. Other than that, the code handling in the assignment for a file remains the same.

This concept allows for easy redirection of data between files and processes, as well as between processes themselves. This concept is an example of **inter-process communication.**

In this example, our program 'dog' opens a file directly and uses its file descriptor:

```
> ./dog file_name
```

In this example, our program 'dog' reads the same data, but through STDIN_FILENO:

```
> ./dog < file_name
```

In this example, we get the process 'ls' to provide the input to 'dog':

```
> ls | ./dog
```