CSE 15L Lab 1

Lab reports for Winter 24' CSE 15L at UCSD.

CSE 15L Lab 1

Jacob Hansen PID: A18031849

cd

`cd` stands for change directory and can be used to move the working directory to the path given in the argument.

1. `cd` with no arguments has no output, and changes the working directory to home. On my machine that is `/Users/zoophere` . The only change is in the working directory displayed.

   Working Directory: `/home/lecture1`

   ```
   [user@sahara ~/lecture1]$ cd
   [user@sahara ~]$
   ```

   Running `cd` with no arguments does not result in an error because running cd with no arguments is a defined part of the command. When looking at `man cd` it says that the behavior of `cd` without arguments depends on if the HOME environment variable is defined. If there is one defined, `cd` with no args just goes to the directory stored in HOME. Otherwise, depending on the system `cd` would just do nothing.

2. `cd ./lecture1/` changes the working directory to the directory passed in the argument, in this case `/home/lecture1` . If the path argument path is relative it works from the current working directory, and if the argument is an absolute path it goes to that directory regardless.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ cd ./lecture1/
   [user@sahara ~/lecture1]$
   ```

   Running `cd` with a valid directory as an argument does not create an error because `cd` is designed to take a directory or path as an argument.

3. `cd ./notes.md` tries to execute `cd` on the file `notes.md` which will throw an error because `cd` is not designed to work on files.

Working Directory: `/home`

```
[user@sahara ~]$ cd ./notes.md
bash: cd: ./notes.md: Not a directory
[user@sahara ~]$
```

Running `cd` with a file as the argument always results in an error because `cd` can't change the terminal's directory to a file. Files can be read, written, or executed with other commands. It doesn't make sense to have a terminals working directory be inside a file.

ls

`ls` lists directory all contents except for hidden files.

1. `ls` with no arguments lists all files or folders in the current working directory to the standard output.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ ls
   lecture1  notes.md  part-two.png test.txt
   [user@sahara ~]$
   ```

   Running `ls` without any arguments is the most common usage for the command, and is not an error.

2. `ls ./lecture1/` lists the directory contents of the argument.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ ls ./lecture1/
   Hello.class  Hello.java  messages  README
   [user@sahara ~]$
   ```

   Running `ls` with a directory as an argument is not an error because it is a well defined behavior for the command. Instead of running `ls` on the current working directory, it runs on the argument directory instead.

3. `ls ./notes.md` writes the name of the file to the standard output.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ ls ./notes.md
   ./notes.md
   [user@sahara ~]$
   ```

This result is not an error, because `ls` is designed to also work on files as well as directories. From the `man` page for `ls`: "For each operand that names a file of a type other than directory or symbolic link to a directory, ls shall write the name of the file as well as any requested, associated information."

cat

`cat` is short for concatenate. It concatenates the content of files onto the standard output.

1. `cat` without arguments results in freezing the terminal so it must be escaped with Ctrl + C.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ cat
   ^C
   [user@sahara ~]$
   ```

   This result could be considered an error because it locks up the terminal. The `cat` command without arguments uses the standard input instead of a file as the input, and when running `cat` alone the standard input is empty which causes the freeze.

2. `cat ./lecture1/` prints a message saying ./lecture1/ is a directory.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ cat ./lecture1/
   cat: ./lecture1/: Is a directory
   [user@sahara ~]$
   ```

   This result is an error because `cat` is designed to output the contents of a file to the terminal. Files and directories have different ways of storing content, so `cat` is not designed to understand the content of a directory.

3. `cat ./test.txt` outputs the contents of `test.txt` into the standard output.

   Working Directory: `/home`

   ```
   [user@sahara ~]$ cat test.txt
   This is a test file for testing `cat`
   [user@sahara ~]$
   ```

   This result is not an error because using `cat` with a file as the argument is the most common usage for the command.