Name: Trevor Thompson

Submit into Canvas, as a MS Word or pdf document by the due date.

Shell Commands

The first command to become familiar with are the commands used to get information about commands. To find out about the `ls` command, you can type “`man ls`”. Once you find what you’re looking for, you can type `q` to quit from `man`.

What do the following commands do? Give a brief description. (Use the `man` pages or just experiment to find out.)

1)	`man`	provide information
2)	`cd`	move directories
3)	`ls`	show contents of current directory (try `'ls -tral'`)
4)	`rm`	remove file
5)	`mkdir`	make directory
6)	`rmdir`	remove directory
7)	`diff`	compare file contents
8)	`echo`	display a line of text
9)	`chmod`	change file mode bits (permissions)
10)	`mv`	move or rename a file
11)	`cp`	copy a file
12)	`cat`	concatenate files and print on the standard output
13)	`less`	“opposite of more” print file one page at a time
14)	`w`	Show who is logged on and what they are doing
15)	`finger`	provide user information (try `'finger rchaney'`)
16)	`history`	display previous commands
17)	`grep`	search and display patterns in files
18)	`exit`	exit shell
19)	`pwd`	display current path
20)	`clear`	clear terminal
21)	`wc`	print newline, word, and byte counts for each file
22)	`seq`	print a sequence of numbers
23)	`ln`	link files together
24)	`time`	track and display file run time

C Programming Functions

What do the following functions do? Give a brief description, identify the include file necessary to call the function from a C program, and write down the return type. (Use the `man`.) There are functions what have the same name as commands. **Be sure you are looking at a C function, NOT a command**.

```
1)  chdir()      _change working directory_ _unistd.h_ _int_
2)  unlink()     _deletes a name from the filesystem_ _unistd.h_ _int_
3)  mkdir()      _make a new directory_ _sys/stat.h_ _int_
4)  chmod()      _change a file's mode bits_ _sys/stat.h_ _int_
5)  fopen()      _Open a file_ _stdio.h_ _FILE *_
6)  fclose()     _Close a file_ _stdio.h_ _int_
7)  open()       _Open and optionally create a file_ _fcntl.h_ _int_
8)  close()      _Close a file descriptor_ _unistd.h_ _int_
9)  printf()     _prints an output_ _stdio.h_ _int_
10) scanf()      _takes an input_ _stdio.h_ _int_
11) fprintf()    _formatted output to a stream_ _stdio.h_ _int_
12) fscanf()     _formatted input from a stream_ _stdio.h_ _int_
13) read()       _read bytes from a file descriptor_ _unistd.h_
    _ssize_t_
14) write()      _write bytes to a file descriptor_ _unistd.h_
    _ssize_t_
15) perror()     _print last error to stderr_ _stdio.h_ _void_
16) fgets()      _read a line from a stream_ _stdio.h_ _char *_
17) strlen()     _get string length_ _string.h_ _size_t_
18) strcmp()     _compare two strings_ _string.h_ _int_
19) strncmp()    _compare two strings up to n_ _string.h_ _int_
20) strcasecmp()      _case-insensitive string compare_ _strings.h_
    _int_
21) strncasecmp() _case-insensitive compare up to n_ _strings.h_
    _int_
22) strcpy()     _copy string_ _string.h_ _char *_
23) strncmp()    _compare two strings up to n_ _string.h_ _int_
24) strncpy()    _copy up to n chars_ _string.h_ _char *_
25) strcat()     _append string_ _string.h_ _char *_
26) index()      _find char in string (first)_ _strings.h_ _char *_
27) rindex()     _find char in string (last)_ _strings.h_ _char *_
```

28) malloc()	_allocate memory_ _stdlib.h_ _void *_
29) calloc()	_allocate zeroed array_ _stdlib.h_ _void *_
30) free()	_free memory_ _stdlib.h__ _void_
31) memset()	_fill memory with a byte_ _string.h_ _void *_
32) strdup()	_duplicate string_ _string.h_ _char *_
33) strfry()	_randomly shuffle string_ _string.h_ _char *_
34) isalnum()	_is alphanumeric?_ _ctype.h_ _int_
35) iscntrl()	_is control character?_ _ctype.h_ _int_
36) isdigit()	_is decimal digit?_ _ctype.h_ _int_
37) isspace()	_is whitespace?_ _ctype.h_ _int_
38) isupper()	_is uppercase?_ _ctype.h_ _int_
39) getopt()	_parse short options_ _unistd.h_ _int_
40) assert()	_abort if condition false_ _assert.h_ _void_
41) strtol()	_string to long_ _stdlib.h_ _long_
42) strtoul()	_string to unsigned long_ _stdlib.h_ _unsigned long_
43) strtof()	_string to float_ _stdlib.h_ _float_
44) atoi()	_string to int_ _stdlib.h_ _int_
45) atoll()	_string to long long_ _stdlib.h_ _long long_
46) time()	_get current time_ _time.h_ _time_t_

Using some Shell Commands

Write down the command and options for doing the following (use `man` to help find answers)

1. List all files, including "hidden" files. ______________ls -a_____________ To search for `ignore` within the `man` page for `ls`, type the following `'/ignore'` and press return.

2. List all files, including their sizes and timestamps. __________ls -al_____________

3. List all files, including their sizes and timestamps sorted so that the newest file is last. _ls -altr_

4. Delete all files in a directory **and** in all subdirectories of that directory

   find Directory -type f -delete

5. Copy all files in a directory **and** all subdirectories to a new location:

   cp -a source/. /path/to/dest/

Make sure you are in your “home” directory (type `cd` and press enter). Typing just `‘cd’` followed by return is like Dorothy clicking her heels together and saying “There’s no place like home.” Use the `pwd` command to see that you are in your “home” directory. This is your **home directory**.

The `mkdir` (make directory) is used to create a new directory. Use this command to create a directory called “cs333” in your home directory.

The `cd` (Change Directory) command is used to change your current directory (`cd cs333`). Use this command to change to your `cs333` directory. Use `pwd` to make sure the `cd` command worked as expected. Create another directory called “`Lab1`” within the `cs333` directory.

What happens when you type `cd` without any parameters? _return to home_

Files have an associated protection (or mode) that limits who can do what with the files. Use the following command to create a file in your `Lab1` directory:

```
echo "stuff" > my.file
```

The > symbol means **redirect the output from the previous command** (in this case `echo`) into the file name that follows (in this case `my.file`).

Yes, that is two greater than symbols.

Add some more text into `my.file` by using this:

```
echo "more stuff" >> my.file
```

The >> symbols means **redirect and append the output from the previous comm**and (in this case `echo`) into the file name that follows (in this case `my.file`).

Show the contents of the file in your terminal:

```
cat my.file
```

Use the `chmod` command to change the mode of the file so that you have full access, people in your group can read the file, and no one else can do anything with it.

What command line did you use? _chmod 740 my.file_

Copy a file from my home directory into your `Lab1` directory. To do this you should enter the command:

```
cp ~rchaney/file.txt .
```

Yes, that is a dot at the end of the command. **It is required.**

The ~ (a tilde) character is a reference to a home directory, in this case my home directory. If you use the ~ alone, without a user log name following it, it means **your** home directory. So,

```
cp ~rchaney/file.txt ~/cs333/Lab1
```

Means copy the file `file.txt` from my home directory to your `cs333/Lab1` directory, under your home directory. Try it.

Final note

The labs in this course are intended to give you basic skills. **In later labs, we *assume* that you have mastered the skills introduced in earlier labs.** If you don't understand, ask questions.