CS 333

Name: Molly Diaz (no name no grade)

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`	opens manual page	
2)	`cd`	change directory	
3)	`ls`	list files	(try `'ls -tral'`)
4)	`rm`	remove	
5)	`mkdir`	make (empty) directory	
6)	`rmdir`	remove (empty) directory	
7)	`diff`	compare files line by line	
8)	`echo`	display strings/files	
9)	`chmod`	change mode bits	
10)	`mv`	move / rename	
11)	`cp`	copy	
12)	`cat`	display file contents	
13)	`less`	low-resource text viewer	
14)	`w`	display user info	
15)	`finger`	fetch info abt user	(try `'finger rchaney'`)
16)	`history`	shows command history	
17)	`grep`	search + regex	
18)	`exit`	exit shell	`(what do you see?)`
19)	`pwd`	print working directory	
20)	`clear`	clears screen	
21)	`wc`	word count	
22)	`seq`	generate sequence	
23)	`ln`	link file (think shortcut)	
24)	`time`	measures exec time of file/command	

Portland State Computer Science

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 dir.	unistd.h	int
2)	unlink()	deletes link file	unistd.h	int
3)	mkdir()	make directory	sys/stat.h	int
4)	chmod()	change file's mode bits	sys/stat.h	int
5)	fopen()	opens file & streams	stdio.h	FILE ptr
6)	fclose()	closes stream	stdio.h	int
7)	open()	opens file descriptor	fcntl.h	int
8)	close()	closes file descriptor	unistd.h	int
9)	printf()	write to stdout	stdio.h	int
10)	scanf()	reads from stdin	stdio.h	int
11)	fprintf()	write to specified stream	stdio.h	int
12)	fscanf()	read from specified	stdio.h	int
13)	read()	attempts to read to bytes	unistd.h	ssize_t (# of bytes)
14)	write()	writes to count bytes	unistd.h	ssize_t (# of bytes)
15)	perror()	prints last error	stdio.h	void
16)	fgets()	gets string from stream	stdio.h	char *
17)	strlen()	returns # of bytes	string.h	size_t
18)	strcmp()	compares 2 strings	string.h	int
19)	str**n**cmp()	compares x bytes	string.h	int
20)	strcasecmp()	compare, ignore case	strings.h	int
21)	str**n**casecmp()	compares x, ignore case	strings.h	int
22)	strcpy()	copies string from s->d	string.h	char *
23)	str**n**cmp()			
24)	str**n**cpy()	copies x bytes s->d	string.h	char *
25)	strcat()	cat s -> d	string.h	char *
26)	index()	returns index of 1st	strings.h	char *
27)	rindex()	returns index of last	strings.h	char *
28)	malloc()	allocate memory	stdlib.h	void *
29)	calloc()	allocate contig. memory	stdlib.h	void *
30)	free()	frees memory	stdlib.h	void

31) memset()	fill memory c byte	string.h	void *
32) strdup()	duplicates string	string.h	char *
33) strfry()	randomize string	string.h	char *
34) isalnum()	tests is alphanumeric	ctype.h	int
35) iscntrl()	tests is control	ctype.h	int
36) isdigit()	tests is number	ctype.h	int
37) isspace()	test is space	ctype.h	int
38) isupper()	test is uppercase	ctype.h	int
39) getopt()	parses commandline	unistd.h	int
40) assert()	abort/error if false	assert.h	void
41) strtol()	convert char -> long	stdlib.h	long
42) strtoul()	convert char -> ulong	stdlib.h	unsigned long
43) strtof()	convert char -> float	stdlib.h	float
44) atoi()	convert string -> int	stdlib.h	int
45) atoll()	convert string -> ll	stdlib.h	long long
46) time()	# seconds since epoch	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 -a -s -t__ 5
3. List all files, including their sizes and timestamps sorted so that the newest file is listed last. + -r
4. Delete all files in a directory **and** in all subdirectories of that directory __rm -r__
5. Copy all files in a directory **and** all subdirectories to a new location: __cp -r__

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? back 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`).

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

Yes, that is two greater than symbols.

```
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 770

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

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

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

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.