LECTURE 1.1: BASIC UNIX (G&A CH. 3)

DEFAULT I/O (INPUT/OUTPUT)

There are 3 default I/O channels that are always assumed active for every command or program:

stdin	standard input where a program expects to find input
stdout	standard output where a program writes its output by default
stderr	standard error where a program writes error messages

By default, all 3 I/O channels are the terminal running the command or program.

MANUAL PAGES

The man utility provides information about a utility.

The manual pages are online copies of the Linux documentation, which is typically divided into 8 or 9 sections.

man [number] [word]	displays the manual entry associated with word, in section [number]. if no number is specified, generally the most commonly used entry is displayed.
man -k [keyword]	displays a list of all manual entries that contain keyword.

more, less

more

more -f [+lineNumber] [filename]*

scroll through a list of files, one page at a time.

+lineNumber	By default files will display from line 1. Use this option to start from lineNumber.
-f	option to not fold long lines
spacebar	display next page
enter	display next line
q	quit from more

less added the capability of forward and backward navigation.

less

By default, man pages are navigated with less.

You can press h in a man page for a summary of less commands. Here are a few useful ones:

space or f or ctrl-f	forward one window
d or ctrl-d	forward (down) half a window
b or ctrl-b	backward one window
u or ctrl-u	backward (up) half a window
/[word]	start search mode for word
p and n	focus previous and next results in search mode

SPECIAL CHARACTERS

special characters (or metacharacters) are interpreted specially when typed in a Linux terminal window.

To see the special characters: $ stty -a
(note that -a option signifies "all")

erase = ^?;	backspace one character
kill = ^U;	erase all of the current line
werase = ^W;	erase the last word
rprnt = ^R;	reprint the line
flush = ^O;	ignore any pending input and reprint the line
lnext = ^V;	don't treat the next character specially
susp = ^Z;	suspend the process for a future awakening
intr = ^C;	terminate (interrupt) the foreground job with no core dump Used to stop a program before it's finished. Some programs are immune to ctrl-c, but most of the time this will kill your process and return you to the shell prompt.
quit = ^\;	terminate the foreground job and generate a core dump
stop = ^S;	stop/restart terminal output
eof = ^D;	end-of-input for utilities that take input from the keyboard, you must use this to signify that the input is finished.

(note that the caret symbol (^) denotes ctrl+)

OPTIONS

options	UNIX allows program options to be specified at the command line. Denoted by a hyphen preceding letters.

pwd

print working directory

shows your shell's location in the directory hierarchy.

PATHNAMES

Two files in different directories may have the same name. They may be unambiguously specified by their pathnames.

pathname	a sequence of directory names that lead you through the hierarchy from a starting directory to a target file.
absolute/full pathname	a pathname relative to the root directory. Example: /home/jmcurran/cs240/subfolder/filename.txt
relative pathname	can also specify file using a pathname relative to current working directory special fields are provided that may be used when supplying a relative pathname: . current directory .. parent directory

Example:
[assume current working directory is "subfolder" from above example and we want to specify "somefile.txt", which is in "cs240"]
../somefile.txt

cat

concatenate

Takes its input from stdin or from a list of files and displays them to stdout.

cat is best for small files; it does not pause between screens of output.

cat [filename]	displays filename on your terminal screen e.g., cat names.txt places.txt use -n option to add line numbers to output e.g., cat -n filename.txt

head, tail

head -n [filename]*	tail -n [filename]*

Displays the first (head) or last (tail) n lines of a file.

If n is not specified, defaults to 10.

If more than one file specified, a small header identifying each file is displayed before its contents.

ls

list

Lists information about a file or directory.

ls -adglsFGR [filename]* [directoryname]*

ls	lists all of the files in the current working directory in alphabetical order, excluding files whose name starts with a period.
-a	includes hidden files (files whose name starts with a period)
ls [filename]	lists info for a specific file (or multiple files)
ls [directory]	lists info for a specific directory (or multiple directories)
-d	list the details of the directories themselves, rather than their contents
-g	lists a file's group
-l	long listing: includes permission flags, file's owner, last modification time.
-s	includes num. of disk blocks the file occupies
-F	causes a character to be placed after file's name to indicate type of file: * executable file; \ directory file = socket; @ symbolic link | FIFO (pipe)
-G	omit group information from listing.
-R	recursively lists the contents of a directory and its subdirectories

LS FIELDS

the ls -la command will return a list with 8 fields (columns), something like:

```
-rw-r--r--  1 jsmith ugrad  512 Oct 31 11:04 file.txt
```

field #	example	description
1	drwxr-xr-x	file type/permissions (see below)
2	1	the hard link count
3	jsmith	username of the owner of the file
4	ugrad	the group name of the file
5	512	the size of the file, in bytes
6	Oct 31 11:04	date and time file last modified
7	file.txt	the name of the file

FILE TYPE/PERMISSIONS

The file type/permissions field can be broken up as follows, from left to right:

The first symbol signifies the file type:

-	regular file	b	buffered special file
d	directory file	c	unbuffered special file
l	symbolic link	s	socket
p	pipe		

The next 9 symbols break into three sets of 3 symbols each:

User (owner)	Group	Others

Types of permission:

-	r	w	x
none (denied)	read	write	execute

Example: drwxr-xr--

file type:	directory
owner permissions: rwx	owner can read, write, execute
group permissions: r-x	group can read and execute
other permissions: r--	others can read only

chmod

change a file's permissions (change mode)

chmod -R [change parameters] [fileName]

Two ways to use chmod:

1. build the change parameters as follows:
   [cluster] [action] [permission]

cluster can be:	u (user/owner)
	g (group)
	o (others)
	a (all)
action can be:	- (subtract permissions)
	+ (add permissions)
	= (assign permissions absolutely)
permission can be:	r (read)
	w (write)
	x (execute)
	s (set user ID / set group ID)

   example:
   remove others' write permission:
   chmod o-w filename.txt

2. specify new permission as an octal number:

rwx is associated with binary numbers	1 for granted, 0 for denied
	e.g., 111 means read, write, and execute permission granted

   the binary numbers are converted to octal numbers.
   e.g., 111 is 7, 101 is 5, 011 is 3, 100 is 4, etc.
   each octal digit represents a permission triplet.

   example:
   we want to set permissions to -rwxr-xr--

rwx = 111 = 7 r-x = 101 = 5 r-- = 100 = 4	chmod 754 filename.txt

The -R option recursively changes modes of the files in directories.

mv

mv -i oldFilename newFilename

Renames oldFilename as newFilename.	Caution: if newFilename already exists, it is replaced.
	the -i option prompts for confirmation if it already exists.

mv -i {fileName1}* directoryName

Move a collection of files to a directory.

mv -i directoryName1 directoryName2

Move directoryName1 into directoryName2.

cp

cp -i oldFilename newFilename

Copies oldFilename to newFilename.	Caution: if newFilename already exists, it is replaced.
	the -i option prompts for confirmation if it already exists.

cp -ir {filename}* directoryName

Copies a list of files into directoryName.

-r	causes any source files that are directories to be recursively copied.

mkdir

make directory

mkdir [-p] newDirectoryName

Creates a new directory. The -p option creates any parent directories in the newDirectoryName pathname that do not already exist.

Returns an error if newDirectoryName already exists.

cd

change directory

cd	changes shell's current working directory to owner's home directory
cd [directoryName]	changes current working directory to directoryName

Note that you can use the . and .. fields with cd.

rmdir

remove directory

rmdir [directoryName]

Deletes directoryName. directoryName must be empty.

Caution: using the rm -r [directory] command recursively removes directory and all of its contents. Be careful with this command!

rm

remove file

rm [name]	deletes name.
-i	option to prompt user for confirmation before deleting name
-r	recursive remove (used to delete nonempty directories)
-f	inhibits all error messages and prompts

lpr

printing a file

lpr [-Pprinter] [-#copies] [fileName]*

-P	specifies the printer (if not included, printer in environment value $PRINTER is used)
-#	prints 1 copy be default, use this option for more