COMMANDS FOR CS 35L

Command	Options	What
pwd		print working directory (absolute path)
cd	~ home . current / root .. parent	change directory
mv		move/rename
cp	-r: recursively copy	copy
rm	-r: recursively delete directory and contents	remove a file
mkdir rmdir		make/remove (empty) directory
ls	-d: directories -a: all files -l: long listing -s: size of file in blocks -t: list according to modification time -i: inode	list contents of a directory
ps		process status, list running processes
kill <pid>		kill unwanted process
wget <url>		Get webpage/file from url
ln	-s: symbolic linking	Links a file, default hard link
chmod	WEEK1	Change permissions of a file or directory
find	WEEK1	Gives location of file in a system
whereis		Locates binary, source, and man page files for a command
whatis		Returns Name section of man page, gives one liner description
touch		Update access/modification time to now
wc	-l: lines -m: chars -c: bytes -w: words	Counts stats of a given file
head	-#: That many lines	Extracts top of files

tail		Extracts bottom of files
man	section: 1: executable programs/shell commands 2: system calls (kernel) 3: library calls etc. -k <word>: print all commands with specific word in man pages	Info about linux command
diff	-i: ignore case -w: ignore whitespace -B: ignore blank lines -u: unified output -s: report when same	compare files line by line
cmp	-b: print differing bytes	compare files byte by byte
comm	-123: reject cols 123 -: takes from stdin	compare two **sorted** files **line by line** depending on locale select or reject lines common to two files
sort	-b: ignore leading blanks -f: ignore case -s: stable sort -u: unique	sorts **lines** of text files depending on locale
tr	-c: complement, tr works on complement of specified chars (NOT) -s: replaces sequence of repeated chars in set1 with single instance of char in set2	translate **or** delete chars, more info in printouts **know what happens with repeating chars in set2 and with set2>set1**
echo	-e: allow escape chars	writes arguments to stdout
printf	#include <stdio.h>	Outputs data with complex formatting
grep	-E (egrep): extended regex (ERE), no backslashes needed -F (fgrep): fixed strings rather than regex	matches **lines**, print lines matching pattern (regex). Uses basic regexs (BRE), meta-characters ?, +, {, |, (,) lose special meaning, instead use backslashed versions
sed	sed ‘s/**regex**/**replacetext**/’	replace parts of text, filter and transform
tar	tar -xzvf filename.tar.gz -x: extract -z: gzip -v: verbose -f: file	Decompress file in current directory

Shell Script	Usage/Reqs	What
set -x set +x		Turn tracing on Turn tracing off
var="hello"	No spaces	Variable declaration
$var_name	echo $var	Variable reference
#		Number of arguments given to current process
?	0 = success > 0 = failure	Exit status of previous command
IFS	Usually newline,tab, space IFS=$'\n'	Internal Field Separator, aka what char separates words
$	$1 … $9 ${10} … ${#>10}	parameters for command-line arguments
$0		Name of script itself
if	if []; then … else … fi	
while	while []; do … done	
for	for f in $tmp; do … done	
''	single quotes	literal meaning, no expansion
""	double quotes	expand backticks and $
`` or $()	backticks	expand as shell commands

declare -a array	arrayname = (element1, e2, e3);
	arrayname = ('e1' 'e 2' 'e 3');
	${#arrayname[@]}: length of array
	arrayname[indexNum]=var
	${array[@]}: reference all items in array
		initialize an array

let count=0		create integer variables

GIT	Usage/Reqs	What
git init	repo creation	start a new repo
git clone	repo creation	create copy of existing repo
git checkout <tag/commit> -b <new_name>		creates new branch
git add		stage modified/new files
git commit	-m “”: message -f Changelog: from file	check-in the changes to the repo
git status		shows modified files, new files, etc.
git diff		compares working copy with staged infos
git log		shows history of commits
git show <hash>		show a certain object in the repo
git checkout HEAD main.cpp	revert	gets HEAD revision for working copy
git checkout -- main.cpp	revert	reverts changes in working directory
git revert	revert	reverts commits (creating new commits)
git reset		deletes commit completely, cleaner
git clean		Cleans up untracked files
git tag	git tag -a v1.0 -m ‘Version 1.0’	human readable pointers to specific commits, tagging
git help		help
git pull		incorporate changes from remote repo into current branch
git format-patch-1		Get patch for last commit

Redirect/Syscalls	Usage/Reqs	What
locale		prints info about current locale env to standard output
export VAR=		change value of an environment variable
stdin	0 #include <stdio.h>	data going into program read in: <&0
stdout	1 #include <stdio.h>	where program writes its output data
stderr	2 #include <stdio.h>	where program writes its error messages
<	program < file	redirects file to program's stdin
>	program > file	redirects program's stdout to file
2>	program 2> file	redirects program's stderr to file2
>>	program >> file	appends program's stdout to file
|	program1 | program2	assigns stdout of program1 as stdin of program2
int getchar()	#include <stdio.h>	Returns next char from stdin
int putchar()	#include <stdio.h> int putchar(int char)	Writes a character to current position in stdout
fprintf	#include <stdio.h> int fprintf(FILE *fp, const char * format, …)	File *fp can be either file pointer or stdin, stdout, stderr. Formatted OUTPUT
fscanf	#include <stdio.h> int fscanf(FILE *fp, const char * format, …)	Formatted INPUT
malloc	#include <stdlib.h> void ***malloc*** (size_t size)	Returns pointer to allocated memory of size bytes
realloc	#include <stdlib.h> void ***realloc*** (void *ptr, size_t size)	Changes size of memory block pointed to by ptr to size bytes
free	#include <stdlib.h> void **free** (void *ptr)	Frees the block of memory pointed to by ptr
time	time <command>	Outputs real, user, sys times Read by clock, CPU time used by process, CPU time used by system on behalf of process
strace		intercepts and prints out syscalls to stderr or to an output file

read	ssize_t read(int fildes, void *bug, size_t nbyte)	Syscall, # include <unistd.h>
write	ssize_t write(int fildes, const void *buf, size_t nbyte)	Syscall, # include <unistd.h>
open	int open(const char *pathname, int flags, mode_t mode)	Syscall, # include <unistd.h>
close	int close(int fd)	Syscall, # include <unistd.h>
getpid	pid_t getpid(void)	Syscall, # include <unistd.h> Returns process ID of calling process
dup	int dup (int fd)	Syscall, # include <unistd.h> Duplicates a file descriptor fd, returns a second file descriptor that points to the same file table entry as fd does
fstat	int fstat(int filedes, struct stat *buf)	Syscall, # include <unistd.h> Returns info about the file with the descriptor filedes into buf
fopen	#include <stdio.h>	Equivalent to syscall open, a library function for file I/O
fclose	#include <stdio.h>	Equivalent to syscall close, file I/O

Parallelism	**Usage/Reqs**	**What**
pthread_create	int pthread_create(pthread_t *tid, const pthread_att_t *attr, void *(my_function)(void *), void *arg)	Creates a new thread within a process and makes executable ● tid = unique identifier for newly created thread ● attr = object that holds thread attributes (NULL for default) ● my_function = function that thread will execute once created ● arg = single argument that may be passed to my_function (NULL if none) 0: success, else error #
pthread_join	int pthread_join(pthread_t tid, void **status)	Waits for another thread to terminate ● tid = thread ID of thread to wait on ● status = exit status of target thread stored in location pointed to by *status. Pass in NULL if status not needed 0: success, else error #
pthread_equal		Compares thread ids to see if they refer to the same thread
pthread_self		Returns the id of the calling thread
pthread_exit		Terminates the currently running thread

GCC/Linking	Usage/Reqs	What
-static	gcc -static hello.c -o hello-static	Static linking
dynamic	gcc hello.c -o hello-dynamic	Default linking is DSO, dynamic shared object
-fPIC		Compiler directive to output **position independent code**, which is required for shared libraries
-l<library>		Link with "lib*library*.a"
-L		Find the library from this path at compile time. Without -L, uses /usr/lib
-Wl,rpath=.:		-Wl passes options to linker, -rpath at runtime finds .so from this path
-c		Generate object code from c code
-shared		Produce a shared object which can then be linked with other objects to form an executable
__attribute__ ((__?__))	__attribute__ ((__constructor__))	Used to declare things about functions called in program (helps optimize calls)
dlopen	#include <dlfcn.h> void *dl_handle; dl_handle = dlopen("libyo.so", RTLD_LAZY); //RTLD_NOW	Makes an object file accessible to a program RTLD_LAZY vs RTLD_NOW ● Resolved later vs resolved now
dlsym	#include <dlfcn.h> void (*myfunc)(int *); myfunc = dlsym(dl_handle, "fcn");	Obtains the address of a symbol within a dlopen-ed object file

dlerror	#include <dlfcn.h>
	//dlopen/dlsym
	char *error;
	error = dlerror();
		Returns a string error of the last error that occurred

dlclose	#include <dlfcn.h>
	dlclose(dl_handle);
		Closes an object file

GDB	Usage/Reqs	What

$ gcc	$ gcc [flags] <source files> -o <output file>
	
	$ gcc [flags] ***-g*** <source files> -o <output file>
		Compiling a program. Use of -g option enables built-in debugging support

$ gdb	$ gdb <executable>
	
	$ gdb
	(gdb) file <executable>
		Specify <executable> program to debug

(gdb) run	(gdb) run [arguments]	Run program
(gdb) quit		Exit GDB debugger
(gdb) break	(gdb) break file.c:6 (gdb) break my_function (gdb) break [pos] if <expression>	Stop the running program at a specific point. Will pause and prompt for command. Can set a breakpoint at a line in a file, at a function, at a position if expression is true, etc.
(gdb) info breakpoints	(gdb) info breakpoints (gdb) info break (gdb) info br (gdb) info b	Get information on breakpoints
(gdb) delete	(gdb) delete [bp_number | range]	Deletes the specified breakpoint or range of breakpoints. Will affect all if no arguments provided.
(gdb) disable	(gdb) disable [bp_number | range]	Temporarily disables the specified breakpoint or range of breakpoints. Will affect all if no arguments provided.
(gdb) enable	(gdb) enable [bp_number | range]	Restores the specified breakpoint or range of breakpoints. Will affect all if no arguments provided.
(gdb) ignore	(gdb) ignore <bp_number> <iterations>	Pass over breakpoint certain number of times before stopping
(gdb) print	(gdb) print [/format] <expression> (gdb) print /d var_name d: decimal x: hexadecimal o: octal t: binary	Prints the value of the specified expression in the specified format

(gdb) c		Continue executing until next breakpoint
(gdb) s		Continue to next source lines, steps IN to functions
(gdb) n		Continue to next source line in current stack frame (do not step into functions)
(gdb) f		Resume execution until current function returns, finish. Execution stops immediately after program flow returns to function’s caller. Function return value displayed.
(gdb) watch	(gdb) watch my_var	Debugger stops program when value of my_var changes and then prints old and new values
(gdb) rwatch	(gdb) rwatch <expression>	Debugger stops program whenever the program reads the value of any object involved in the evaluation of expression
(gdb) backtrace		Shows the call trace/call stack, gives frame listing including arguments in function and line being executed
(gdb) info	(gdb) info frame (gdb) info locals (gdb) info args (gdb) info functions	Displays info about current stack frame (return address and saved register values) Info about local variables and values, argument values, functions
(gdb) list		Lists all source code lines around current line
Tab		Autocomplete
Up/Down		Command history
help <command>		More info about a command