Unix

Use of the Unix System on a Network

`man` – read a manual entry
`man ls` – gives you the entry for `ls`
`man 2 write` – section 2 entry for `write`
`man -k disk` – lists all manual entries whose summaries include the work *disk*

Note: Manual section 1 contains commands. Sections 2 and 3 contain the API calls.
As programmers, we usually want the section 2 or 3 entry,
it gives the parameters and includes for the system calls.

Warning: machines differ (IBM vs. Linux vs. Free BSD)
Read the manual for the correct machine

Linux pseudo terminals:
you have 6 terminals, alt-F1 .. alt-F6
you can have up to 6 logins from one keyboard
don't forget to logout on all terminals you were using

Network debugging techniques:
Run the client in one virtual terminal and the server in another.
Have terminals open (using telnet) on different machines.

Compilers

`gcc` – the standard c compiler

Warning: what you compile on one machine will not run on another.

`.o` files created by one compiler cannot be used by another.

Technique: fully recompile, leave no `.o` files.

Example:

```
gcc x.c y.c z.c -o prog
```

Compiles 3 files, calls the executable `prog`
One file must contain a function called `main`
We will not use `.o` files as shown in the book because this is dangerous when compiling for multiple architectures.
If you omit the "`-o`" option the executable will be called `a.out`; dangerous if you are compiling both a client and a server.

To run the program, just type it's name.
Example: `prog`

Copying Files

`cp` – copy file or files

`cp` list_of_files destination
if the destination is a directory,
–the files are placed in the directory
–keeping their original names
if the destination is not a directory,
–the "list" must contain only one file
–and "destination" is the new name of the file
–the file is given the

The wild card (*) and the home directory (˜) symbols may be to specify the list of files. For example:
`ch*t`
specifies all files starting with `ch` and ending with `t`.
`~/y.c`
specifies the `y.c` program from your home directory
`~joe/y.c`
specifies the `y.c` program from joe's home directory

```
cp ~volper/src/*.c ~/
```
copies all C programs from the `src` subdirectory of the instructor into your home directory.
(The final / is optional in this case.)

Printing

The command to print is `lpr`.
Arguments: a list of files to print

Example: `lpr x.c y.c`

Printing is enabled from named machines (i.e., not `lab35`, `lab76`) until the CECS 476 students complete the assignment in which they enable network printing.

X Windows

They may or may not be available depending on how well the student administrator configures it on your machine.

Editing with vi

`vi` the standard unix screen editor.

You can edit text files.
You cannot edit directories or compiler output.
Use any editor you want, I'll review this one here.

Command:

```
vi file_name
```

`file_name` is the name of a file

Action:

The file exists;
you can change it

The file does not exist;
the file will be created.

Example:

```
vi sample.note
```

Editing Concepts

Current location:

where the cursor is

where changes occur

Modes:

insert mode: what you type is inserted into the file

command mode: what you type is an editor command

In command mode: most of the 52 letter keys (upper and lower case) and most of the special keys are commands.

vi Commands
Changing the current location

Incrementally changing the current location: (two ways)

Use the arrow keys.

Use `h`, `j`, `k`, `l`

left, down, up, right.

Moving by jumping `G`

Example: `34G`

jump directly to line 34

vi Commands
Inserting

(two ways to start)

“`i`” insert in front of cursor position

“`a`” insert (append) after cursor position.

both: you enter insert mode.

In insert mode: what you type is inserted.
includes “returns”.
backspace (sometimes the delete key) back up (uninsert).

To “finish” an insertion type `<escape>`
you go back to command mode.

Warning: some versions of vi do allow you to move around using the arrow keys while in insert mode. If the arrow keys produce funny results you are probably still in insert mode on one of these versions.

vi Commands
Deleting

“`d`” followed by how much

“`d2d`” delete 2 lines

the current line and the next line

“`dd`” same as `d1d`.

“`x`” delete a letter

the letter the cursor is on

“`X`” delete a letter

the letter in front of the cursor

vi Commands

Undo: (in case of mistake)

“`u`” undo the last command.

“`U`” undo this line.

The last sequence of commands on this line are undone.

A second

“`u`” undo the undo.

vi Commands

Leaving the editor

"`:wq`" save the changes and exit the editor.

"`:q!`" exit the editor *without* making any changes to the original file. (if you really goofed up)

Saving without leaving the editor

"`:w`" save the changes and stay in the editor.

Trick:
"`:w` *FileName*" save the changes into the specified file

Homework Hint

The flags of interest for a file open are:

`O_RDONLY`	read only
`O_WRONLY`	write only
`O_RDWR`	read/write only (update)
`O_CREAT`	If it doesn't exist, create it
`O_TRUNC`	erase anything that is already in the file

The modes of interest are (this are bits):

`S_IRUSR`	user can read
`S_IWUSR`	user can write

If the file is created, then the mode bits apply and are used to set the read/write/execute attributes of the file. Note: to get read and write you need to "or" the mode bits:

```
S_IRUSR | S_IWUSR
```