COMPUTER SITE GUIDE
Department of Linguistics

James A Harmon
Matthew R Hyclak
support@ling.ohio-state.edu
Department of Linguistics
The Ohio State University
Columbus, Ohio
November 2001

Section 1
Introduction To UNIX

1.1 Login And Logout

Commands discussed in this section are:

- **logout** - log off system
- **ls** - list the contents of a directory
- **passwd** - change password

1.1.1 Logging In

If are sitting at the unix workstation then you will be logging in using a *Graphical User Interface (GUI)*. Enter your *username* and *password* to login. If you are logging in using a PC or Macintosh in the department or from home then you will be logging in remotely and you should see the next section.

1.1.2 Logging In Remotely

You will need an ssh (secure shell) client in order to login remotely. All the computers in the department already have ssh installed.The program SSH Secure Shell for Windows is available on OSU's software to go website (http://softwaretogo.osu.edu). NiftyTelnet for MacOS is available on our ftp site at the URL ftp://ftp.ling.ohio-state.edu/pub/apps/ssh/. For linux you can install OpenSSH available from www.openssh.com. The hostname of the machine you will be connecting to is "ling.ohio-state.edu". In order to use these programs you need to have a computer that is already connected to the internet. The Ohio State University maintains a pool of modems called HomeNet with the following phone numbers:

722-9800 - 30 minutes limit
722-9900 - 6 hour limit

The HomeNet lines require your OSU ID and password. This is different from your linguistics account and you can find information about how to enable your OSU account at the URL http://www.oit.ohio-state.edu/userpass.html.

At the **login:** prompt enter your *username* and then you'll be prompted for your *password*. A typical login session looks like this:

```
login:username
password:password

Last login:  Mon Sep 23 07:24:40 from julius.ling.ohio-s
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Welcome to the Sun Workstations of the Linguistics Department
**** Please mail system questions or comments to ``support''
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
You have mail.

julius:~[1]
```

This last line is called the Unix Prompt, or Shell Prompt. This is where you will enter the commands you would like the computer to perform. The usual format of this prompt is

hostname:*current directory*[*command counter*]

The *hostname* is the name of the machine you are logged into. This could be a machine you are sitting at or, if you connected remotely, the machine you are connected to. The *current directory* is the directory you are presently working in. Lastly, the *command counter* is just a count of how many commands you have entered. After each new command, it is increased by one.

When you enter your password it will not appear on the screen for obvious security reasons. After you enter your *username* and *password* the **message of the day** is displayed. Please read it since important notices and downtimes will be posted here. The system then notifies you if "You have mail" or "You have new mail". The very first thing you should do when you login for the first time is change your password. Use the command **passwd** which will ask you to enter the old password and then the new password twice just in case you mistyped it.

When you login you are in what is called your *home directory*. Typing the command **ls** will display the files and directories in the directory you are in. There are also some special files in your *home directory* that begin with a ".". These are referred to as your "dot files" or "environment files". Since these files begin with a "." they do not appear when you do a normal **ls**. Use **ls -a** to see all files including "dot files". Some of the important environment files are:

.login - contains commands that are executed upon login
.cshrc - contains environment specification and aliases
.logout - contains commands executed upon logout
.emacs - contains specification for the unix editor emacs

DO NOT delete or edit these files unless you know what you are doing.

1.1.3 Logging Out

If you logged in remotely type the command **logout**. If you logged in using the Graphical User Interface then you will need hold down the right mouse button while the mouse is in the background and select "log out" or "exit".

1.2 Files

Commands discussed in this section are:

- **mv** - move or rename files
- **cp** - copy files
- **rm** - remove files

To copy the contents of an existing file to a new file use the command:

`cp` *existing-file new-file*

To rename an existing file to a new file name use the command:

`mv` *old-name new-name*

To move files to a new directory use the command:

`mv` *filename* `directory/`

For example, assume you have a directory called "Thesis". To move the file named "example" to this directory you would type:

```
mv example Thesis/
```

If you want to move the file "example" to a new name in the "Thesis" directory you would type:

`mv example Thesis/`*newname*

To remove files use the command:

`rm` *filename*

All these commands can be run in interactive mode where it will ask for conformation for each file by using the "-i" option. For example:

```
rm -i filename
```

If you use this option you will be asked to verify the removal of a file. In the case of **mv** and **cp**, if *new-filename* already exists you will be asked if you really want to overwrite it. Some of these commands may by default run in interactive mode, this is determined by the environment files given with your account.

1.3 Viewing Files

Commands discussed in this section are:

- **cat** - display a file
- **more** - display a file one page at a time
- **less** - display a file one page at a time

There are several commands that will display the contents of a file on the screen. The simplest command is **cat**. Typing:

```
cat filename
```

will list the entire file on the screen. This command is good for short files but long files will simply scroll the screen and you will only be able to view the last screen of text. The command **more** will display files one screen at a time. Typing:

```
more filename
```

will display the first screen with a percentage (which is the amount of the file that has been displayed) at the bottom. Typing *return* will advance the display one line and typing the *space* key will advance the display one screen. Typing "q" will exit **more**.

A command similar to **more** is **less**. When you view a file using **less** it displays the percentage, the filename and the line number of the first line on the screen. As with **more**, *return* will advance one line and *space* will adavance one screen. In addition you can use **emacs** movement commands to move around the file. For example, C-p will move back one line.

The *less* command also has search commands. To search forward for a string type "/" and then enter the string ending with a *return*. To search backwards type '?' and then enter the string ending with a *return*.

1.4 Directories

Commands discussed in this section are:

- **cd** - change directories
- **mkdir** - make a directory
- **rmdir** - remove directories
- **pwd** - present working directory

The UNIX file system is a hierarchy of directories starting from the root directory, represented with a "/". Typing the command **pwd** will display the path to the current directory. If you type **pwd** in your *home directory* it will display the full path to this directory.

Using the **cd** command you can move around the file system. To change to the /tmp directory, for example, use the following command:

```
cd /tmp
```

To return to your *home directory* you type **cd** without an argument. Another way to return to your *home directory* is to type the following command:

```
cd ~
```


the ~ is shorthand for the full path to your **home directory**. Now, assume you have a subdirectory named Mail in your **home directory**. If you are in your **home directory** you can change to the Mail subdirectory by typing:

```
cd Mail
```

Notice there is no / in front of the argument Mail. If the argument was /Mail the command would mean to change directory to the directory Mail located in the root(/) directory. This is the difference between *absolute* and *relative* paths. If the path begins with a "/" then it is an *absolute* path. A path that does not begin is with a "/" is *relative* because it is relative to the current directory. Another useful fact is that the directory immediately above the current directory is represented by ".." From the Mail subdirectory, the command:

```
cd ..
```

would change directory back to the *home directory*.

In your home directory you can create a hierarchy of directories to organize your files. You can create multiple levels of directories within directories. Each directory can contain any number of files or directories.

1.4.1 Creating directories

To create a directory use the following command:

```
mkdir directory-name
```

1.4.2 Removing Directories

In order to remove a directory it must be empty. In the directory immediately above the directory you want to remove use the following command:

```
rmdir directory-name
```

1.4.3 Renaming Directories

Renaming directories is the same as renaming files. Use the following command:

```
mv old-name new-name
```

Now for an example. Assume you are in your *home directory* and you make the following directories:

```
mkdir Thesis
mkdir Thesis/version1
mkdir Thesis/version1/notes
mkdir Thesis/version2
```

You can then change directory to Thesis/version1/notes with the command:

```
cd Thesis/version1/notes
```

and then to change directory to Thesis/version2 with:

```
cd ../../version2
```

or

```
cd ~/Thesis/version2
```

The first command takes advantage of ".." and the second command uses ~ which, as previously mentioned, stands for your *home directory*. Now the command:

```
cd ..
```

will change directory to Thesis. You could then use the command:

```
rmdir version2
```

to remove the version2 directory. For protection, however, you can only remove a directory if it is empty.

1.5 Wildcards

A useful feature of the UNIX system are wildcards. Wildcards are used to perform a command on multiple files. For example, if you have 50 files in a directory that you want to copy them to another directory it would not be practical to copy them one by one. Wildcards make problems like this trivial.

The two most common wildcards are:
? match any single character
***** match any sequence of characters

For example, the command:

```
rm paper?
```

will remove files that begin with "paper" followed by one more character. It would match "paper1", "papera", "paper+", ect.

As example for the * wildcard, the command:

```
rm *.out
```

will remove any file that ends with ".out". It would match "prog.out", "final.out", "a.out", etc. You can also use multiple wildcards in the same command. For example, the command:

```
rm *paper?
```

will remove files that end with "paper" followed by one more character. It would match "firstpaper1", "secondpaper2", etc.

As further example, to copy or move multiple files to a new location you can type:

```
cp *.c Save/
```

would copy all files that end with ".c" to the directory "Save".

1.6 Getting Help

The UNIX programmer's manual is on-line. For any UNIX command type **man command** to get complete information. For example:

```
man ls
man rm
```

1.7 Printing

Commands discussed in the section are:
- **lpr** *filename* - print a file
- **lpq** - show the printer queue
- **lprm** - remove a print job from the queue

1.7.1 lpr

The **lpr** command sends a file (a regular text file or a PostScript file) to the printer. The **lpr** stands for **l**ine **pr**inter. Note that if you are using TeX you will want to use a different command to print dvi files (see the section on TeX). The command:

`lpr` *filename*

will send the file called *filename* to the default printer.

You can specify which printer you want to print to by explicitly giving the printer name after the **-P** option. For example, to print to the printer named *printer021* you would type:

`lpr -P`*printer021 filename*

1.7.2 lpq

When you print a file it puts the file in a "print queue" so that it can print out requests from different people in an orderly manner (It wouldn't be very useful if it printed half a page of your file and then printed half a page of someone else's paper on the bottom). The print queue works like a line at the grocery—first come, first served (with the exception of system administrator types who can cut in line to the detriment of us all). You can examine what printing jobs are in a print queue by using the **lpq**. This will give a list of jobs, if any, that are to be printed out. The `lpq` stands for **l**ine **p**rinter **q**ueue. This list contains the information: who is printing the job, the identification number the computer gives to that job (which will be used for the next command, **lprm**), and the size of the job in characters (bytes). It can be useful to examine this before you print out a job so you can see if many jobs are already printing (in which case you might want to print it later).

You can get the print queue for a particular printer the same way we did above with the **lpr** command—by using the **-P** option. The following command gets the print queue for the printer named *printer021*:

`lpq -P`*printer021*

If a printer runs out of paper or has a paper jam don't think that the computer will forget to print out your job. As soon as the problem is fixed, it will print out the jobs in the queue until it has another problem (and when this is fixed it will continue to print out the jobs!). You can more intelligently control printing out by using the next command.

1.7.3 lprm

You use this command to take jobs off the print queue (cancel printing them). The `lprm` stands for **l**ine **p**rinter **rem**ove. Say you send a job to the printer, and then you decide that you don't want to print it out. You can remove **all** your jobs on a print queue with the command

```
lprm -
```

You can also specify a particular printer for canceling jobs by saying:

`lprm -P`*fridge* `-`

Or you can cancel particular jobs by doing an `lpq`, seeing what identification numbers have been given to your jobs, and removing these jobs. For example,

```
julius:~[1] ls
  lp.info   emacs.tips   lp.info   ps.info       fortran   latex.help
  lp.tex    ftpstes      lp.log    suggestions   lp.ps     tcp-ip-admin
julius:~[2] lpr lp.info
julius:~[3] lpr latex.help
julius:~[4] lpq
  lp is ready and printing
  Rank   Owner      Job  Files                                 Total Size
  active joe        69   c.4, c.5                              480304 bytes
  1st    jim        70   standard input                        4079 bytes
  2nd    john       71   standard input                        4838 bytes
  3rd    al         72   standard input                        6361 bytes
  4th    xl         73   standard input                        9404 bytes
  5th    jimbob     74   lp.info                               3950 bytes
  6th    jimbob     75   latex.help                            3534 bytes
julius:~[5] lprm 74 75
  dfA074julius dequeued
  cfA074julius dequeued
  dfA075julius dequeued
  cfA075julius dequeued
julius:~[6] lpq
  lp is ready and printing
  Rank   Owner      Job  Files                                 Total Size}
```

```
   active joe        69   c.4, c.5                    480304 bytes}
   1st    jim        70   standard input              4079 bytes}
   2nd    john       71   standard input              4838 bytes}
   3rd    al         72   standard input              6361 bytes}
   4th    xl         73   standard input              9404 bytes}
julius:~[7]
```

1.7.4 Printer Etiquette

Just remember that there are other people using the machine, so try not to put jobs in the queue which will take a lot of time when other people also are printing files. Remember to delete jobs in the queue when you decide you don't want them, instead of letting them get printed out, wasting a lot of time in the process, and then just sitting around the printer room cluttering things up. Also, clean up after yourself, don't leave any junk you printed out just sitting around—recycle it or throw it away.

1.8 Input/Output Redirection

Commands discussed in this section are:

- `date`
- `ls`
- `more`
- `>, >!, >>, >>!, <`

When a program runs on a UNIX system it usually expects some input and produces some output. By default, a program will use the **standard input** which is usually the keyboard and the **standard output** which is usually the screen. For example, the command

```
date
```

has no input and writes the current date and time to **standard output**, which is the screen. The "`>`" command is used to redirect the standard output. For example, the command

```
date > filename
```

redirects the standard output to the file named *filename*. In *tcsh*, if the file to the right of "`>`" already exists then the command aborts and tells you that the file already exists. The *bash* shell will overwrite it by default. There are several variations to redirecting the standard output. In *tcsh*:

`>`*file* - redirect output to *file*, abort if *file* already exists.
`>!`*file* - redirect output to *file*, overwrite *file* if it already exists.
`>>`*file* - append output to *file*, abort if *file* does not exist.
`>>!`*file* - append output to *file*, create *file* if it does not exist.

For example, in *tcsh* to append the contents of file *file1* to the end of file *file2* you would use the "`>>`" redirection command as follows:

```
cat file1 >> file2
```

The "`<`" command is used to redirect the standard input. Redirecting standard input is not used frequently since most UNIX commands take their input from the file (or files) listed on the command line. An example of how input redirection can be used is to e-mail a file called *mailfile* using **pine**. The command

```
pine someone@ling.ohio-state.edu < mailfile
```

mails the contents of the file to the user called `someone`. (See Section 3for a description of *elm*.) Without the redirection command the user would have to type the message from the keyboard.

Another type of redirection is called *piping*. *Piping* is when the output of one command is redirected as the input to another command. The piping command is a "|" (i.e., a vertical bar). For example, the command

```
date | lpr
```


sends the output of **date** to the input of **lpr** (and so prints out the current date and time). It is important to note that a command must always follow a "|". That is, you cannot use *piping* to redirect the output to a file (that is the function of the "`>`" operator).

The above examples are very simple and do not give a full range of the possible uses for these redirection commands. A more realistic example is the following:

```
ls -alt
```

produces a long listing of the current directory with the files listed in the order of time modified (so the most recently created or modified files will be first). If there are more files than there are lines on the screen the first files will scroll off the screen. The command

```
ls -alt | less
```

redirects the listing into `less` and so allows you to view the listing one page at a time.

1.9 File Access Permissions

Commands discussed in this section are:

- **chmod** - change the permission mode of a file
- **ls** - list the contents of a directory

1.9.1 General Overview

All files and directories under UNIX have associated access permission that can be individually set. There are three types of access : *read*, *write* and *execute*. For files, the permissions are straightforward. *read* permission allows viewing the contents of file, but not modifing it. *write* permission allows viewing and modifying, and *execute* permission allows execution of a program. For directories there is one distinction worth noting. In order to **cd** into a directory you must have *execute* permission. In order to see the contents of a directory you must have *read* permission. Each file (or directory) has three sets of permissions. The first set applies to the user (or owner) of the file. The second set applies to those that have group ownership to the file and is referred to as group. The last set applies to everyone on the system and is usually referred to as other.

1.9.2 Using ls -l to view file permissions

ls -l is used to display the current access permissions of a file or directory. For example, assume there is a file named "example" in the current directory and the following command is executed:

```
julius:~[1] ls -l example
-rwxr--r-- 1 jimbob Jan 2 1990 example
```

The ''`-rwxr--r--`'' is the access permission information. The first position tells you if it a file or directory. A "-" means it is a file and a "d" means it is a directory. In this case it is a file. The next three positions are the access permissions for the user(or owner) in the order: read(r), write(w), and execute(x). Then, in the same order, follows the access permissions for the group and then the access permissions for other. For each position, either the appropriate letter is printer is that type of access is allowed or a "-" if it is denied. In this case the user has read, write and execute permission; the group has only read permission and other has only read permission. You may also want to know the group the file belongs to and you can add the "g" option to the *ls -l* command to by typing:

```
julius:~[1] ls -lg example
-rwxr--r-- 1 jimbob staff Jan 2 1990 example
```

This shows the permissions and that the file is owned by user jimbob and group staff.

1.9.3 Using chmod to change access permissions

The command **chmod** can be used to change the access permissions of a file or directory. the syntax for this command is:

chmod *mode filename*

The mode of each file is changed according to *mode*, which may be absolute or symbolic as follows. An absolute mode is an octal number obtained from the following modes:

400 Read by owner.
200 Write by owner.
100 Execute by owner
040 Read by group.
020 Write by group
010 Execute by group
004 Read by others.
002 Write by others.
001 Execute by others.

To determine the *mode* simply add the numbers of the permissions wanted. For example, to give read,write and execute by owner and only read by group and other *mode* would equal 744. The command to set these permissions on file would then be:

`chmod 744` *filename*

The *mode* can also be represented symbolically. The syntax on the symbolic mode is:

kinds-of-users + or - *kinds of access filename*

where *kinds-of-users* is:
u - user(owner) of the file
g - group
o - other
a - all(user,group and other)

and *kinds-of-access* is:
r - read
w - write
x - execute

For example, to give group and other read, write and execute permission, the command would be:

`chmod go+rwx` *filename*

To take away read permission from group and other the command would be:

`chmod go-r` *filename*

Section 2
Emacs

Emacs is a text editor on the UNIX system. This section is intended as a quick introduction and reference guide. In this section we will simply discuss some of the most important commands. A lighter-weight editor that can use these same keybindings is installed on our system called **jed**.

The following notation is used throughout this section:

C- means hold down the control key and type the following character. For example, C-x means hold down the control key and type "x".

M- means hold down the meta key and type the following character. The meta key is usually on the bottom left of the keyboard next to the spacebar.
This will only work on keyboards with a meta key. However, on *all* keyboards you can use the ESC key. That is, hit the ESC key (DO NOT hold it down) and then type the following key.

It is important to realize that **emacs** does not have separate modes for inputing text and for operating on this text (as **vi** does). In **emacs** all printing characters that are typed appear in the text and all commands begin with a non-printing character. For example, if you type `k` this "k" will appear in the text directly under the cursor (the blinking rectangle that indicates where the next character will be printed). If you type `C-k` the remainder of the current line will be deleted. And, if you type `C-x k` (first type `C-x` and then type `k`), the file you are presently working on will be erased from **emacs**' memory!

There are three very important commands to remember as you learn **emacs**. The first two will allow you to undo many of your mistakes:

C-g Quit. Cancel running or partially typed command.

C-_ Undo the last input or the last change (if this command is repeated it will undo each previous input or change in turn).

C-l Refresh the screen.

and the third command exits **emacs**:

C-x C-c Exit **emacs**.

This exit command asks you, in turn, if you want to save each of the files that has been edited during this **emacs** session. If you do not save a file then any changes made since the last time the file was saved (see the subsection on Saving Files and Backup Files) will be lost! The only time when this is the correct choice is if you have made a complete mess of the editing session and want to "kill" all the changes you have made. In this case type `n` and then type `yes` when **emacs** asks you if this is REALLY what you want to do.

2.1 Starting and Stopping Emacs

The normal way to enter the **emacs** editor is to type `emacs` *filename* where *filename* is the name of the file you want to work on. If this file already exists, it will appear on the screen (beginning at the first line of the file). On the other hand, if this is a new file, it will immediately be created and the screen will be blank (because there is nothing in the file yet).

When you work on a file in **emacs**, that file is stored in a "buffer" inside **emacs**. It is possible to work on many files at one time in **emacs**, in which case each file has its own buffer, but only one buffer (and thus one file) appears on the screen at a time (but see the subsection on Multiple Windows for putting more than one file at a time on the screen). As you work on the file the changes you make are stored in the buffer and not in the file on disk.

When you enter **emacs** there are actually *two* different windows that appear. The first contains the current buffer and the second, which is only one line high and is at the bottom of the window, shows the current **emacs** command you are typing (if, in fact, you are entering an **emacs** command and if you wait for a second or two). This is called the "minibuffer". If the current **emacs** command requires further input from you, this will also appear in the minibuffer.

There is a very important fact that must be kept in mind. It is possible to enter **emacs** without specifying a file by simply typing `emacs`. You can then create a new file by merely inputting text. However, this new buffer is not yet connected to a file on disk and so cannot be saved back to disk! If you exit **emacs** without saving this buffer, it will be lost. The command **C-x C-w** (discussed below) can be used to save this buffer to a new file on disk.

2.2 Saving Files and Backup Files

The changes you make to a file are stored in the buffer and are only written out to disk when either:

(i) you give the **emacs** command to save the changes,

(ii) **emacs** automatically saves the changes (which it does every once in a while), or

(iii) you exit **emacs** and save the changes. However, **emacs** automatically creates a backup file whenever a file is first changed. Thus, if **emacs** modifies the file *filename.c* then a copy of the original file is stored in the file *filename.c~* and all the changes in the file (say *filename.c*) can be killed after exiting **emacs** by typing

```
mv filename.x~ filename.c
```

If **emacs** is aborted for any reason it will store the changed files. For example, the changes in the file *filename.f* will be stored in the file *#filename.c#*. When *filename.c* is next opened by **emacs**, if there exist a *#filename#* that is newer than *filename.c*, **emacs** will tell you

```
Auto save file is newer; consider M-x recover-file.
```

You can then use the command **M-x recover-file** *return* to recover the file.

2.3 Files

C-x C-f* Read a file into an **emacs** buffer (it also appears on the screen).
This command will also create a new file if the file does not already exist.

C-x C-s Save the current buffer back to disk (this saves all changes to the file).

C-x C-w* Save the current buffer back to disk *under a new name* and also change the name of the current buffer to this new name.

C-x i* Insert contents of another file into the current buffer at the current cursor position.

NOTE: The * means you will be prompted for further information in the minibuffer and you must end this information with *return*.

2.4 Moving The Cursor

C-b Move the cursor back one character.

C-f Move the cursor forward one character.

C-n Move the cursor down to the next line.

C-p Move the cursor up to the previous line.

C-a Move the cursor to the beginning of the line.

C-e Move the cursor to the end of the line.

C-v Scroll forward one screen.

M-v Scroll backward one screen.

M-< Go to beginning of file.

M-> Go to end of file.

M-x goto-line *return** go to the given line number (the hyphen is part of the command).

2.5 Search And Replace

In **emacs** you can use the following commands to find and replace string patterns. When you use one of these commands, a prompt will appear in the minibuffer asking you for further information. This information must be ended with a *return*.

C-s* Incremental search forward.

C-r* Incremental search backward.

Incremental search means that it will search for the characters as you type them. For example, if you typed "find" it would first find the next "f", then it will find "fi" followed by "fin" and finally "find". If then you use the DELETE key to delete the "d" in "find" the cursor will move back to the first occurence of "fin" (which might be, for example, in the word "final").

To continue searching forward (backward) keep typing `C-s` (`C-r`). After you exit a search, you can search for the same string again by typing just `C-s C-s` (`C-r C-r`). If the string is not found, **emacs** responds `Failing I-Search`; and, if you type `C-s` (`C-r`) again, the search continues at the top (bottom) of the file.

M-x replace-string *return** Replace every occurence of an *old string* with a given *new string* in the rest of the buffer.

M-%* Ask if every occurence of an *old string* should be replaced with a given *new string* in the rest of the buffer.
Replies:
SPC - yes, replace this string and continue to the next occurence.
DEL - no, do not replace this string but continue to the next occurence.
ESC - no, do not replace this string and quit.
! - replace this string and all following occurences without asking again.

M-% is very useful because the user can check if the replacement is correct and try again if it is not.

2.6 Deletions and Cut And Paste

In **emacs** you can use the following commands to delete one character at a time.

DEL Delete the last character (i.e., to the left of the cursor).

C-d Delete the present character (i.e., under the cursor).

Note that the *return* at the end of each line is a character and can be deleted by any deletion command. This is the normal way to join to lines together into one line.

It is also easy to delete larger amounts of text at one time (called "killing" rather than "deleting"). When this is done the material killed is saved for later use (in the "kill" buffer) and can be inserted anywhere else in the text by moving the cursor to the desired point and typing `C-y`. There are a few commands that kill a specified amount of text:

C-k Kill from the cursor to the end of the line.
However, if the cursor is already at the end of the line it only kills the *return* at the end of the line.

M-d Kill the next word.

M-z *char* Kill all the text to the next occurence of the character *char*.

Another common method of killing text is to define a "region" and then kill all the text within that region. A region is defined to be all the text between the cursor and the last "mark" that was set.

C-SPACE Set the mark at the current position.
C-@ Also sets the mark at the current position.
C-x C-x Switch the mark and the cursor (to check what the bounds of the region are).
C-w Kill the region.
M-w Do not kill the region (but still store it in the kill buffer).
(This command allows you to copy text with one command rather than with two, i.e., `C-w C-y`).

The last item in the kill buffer is inserted in the text at the cursor. The kill buffer actually saves the last 16 items of text killed and all of these are accessible by typing `C-y M-y ...`.

C-y Yank last killed text (i.e., put it back in the text at the present cursor location).
M-y This must follow *immediately* the **C-y** or another `M-y` and replaces the text just previously yanked with the preceding item in the kill buffer.

Thus to replace the last item in the kill buffer type `C-y`. To replace the second-from-the-last item type `C-y M-y`. To replace the third-from-the-last item type `C-y M-y M-y`, etc.

It is important to realize that each kill command usually pushes a new item onto the kill buffer. *However*, if two or more of these commands occur in a row (i.e., with *no* keystrokes in between) they combine their text into a single item in the kill ring. For example, if the cursor is at the beginning of a paragraph that is three lines long, then `C-k C-k C-k C-k C-k C-k`, i.e., 6 `C-k`'s in a row, will kill the entire paragraph and put it into the last item in the kill buffer. (Recall that the first `C-k` kills the line of text and the second *C-k* kills the *return* so it takes *two* `C-k`'s to kill an entire line of text.

It is also important to realize that there is only *one* kill buffer. As we will discuss in the next subsection you can kill (or copy) text in one buffer and yank it into another buffer. This is the standard way to move (or copy) text from one file to another.

2.7 Multiple Windows

When you first run **emacs** you open one window and can only work on one file at a time. However, it is possible to work on two parts of the same file (for example to move — or copy — text from one part to the other) or to work on two different files at the same time. Recall that the text you are editing in **emacs** resides in an object called a buffer (and that the changes you make in **emacs** are made to the buffer and not to the file on disk). Each time you visit a file, either by entering **emacs** with a specified file (i.e., typing `emacs` *filename*) or by typing `C-x C-f` *filename* while *in* **emacs**, a new buffer is created to hold the file's text. At any time, one and only one buffer is selected (called the current buffer). The cursor is always connected to the current buffer. If there are two or more buffers in use at the same time you can change the current buffer by typing `C-x b` followed by the *filename* of the desired buffer. By typing `C-x 2` you split the one window into two windows — each of which will show the same text. You can move each window to show a different part of the same buffer; then any change in one window changes the contents of the buffer and so the change also applies to the other window. (This makes it easy to copy or move text from one part of a file to another.) However, if each window is showing a different buffer then each window is completely independent. Text can still be copied or moved from one buffer to the other by killing and yanking. To do this you would split the screen into two windows and then either load another file in one of the windows (with `C-x C-f`) or change the buffer in this window (with `C-x b`). You can then move between the buffers with the **C-x o** command.

C-x 2 Split the window into 2 windows.
C-x o Switch cursor to another window (if more than one exists).
C-x 0 Delete this window (but the buffer still exists).
C-x 1 Delete all windows but this window (again all the buffers still exist).
C-x b* Work on a different buffer in this window (the buffer must already exist).
Note that this cannot be used in place of **C-x C-f** (which creates a new buffer from a disk file).
C-x C-b Open a new window showing all the buffers which currently exist.
C-x k Kill a buffer (the disk file is not killed but it is no longer in a buffer).

2.8 Non-Printing Characters

In **emacs** it is easy to input most characters and to see most characters on the screen. One character that is a problem is ⟨TAB⟩, the TAB character. It can be input by hitting the TAB key. However, ⟨TAB⟩ character cannot be seen on the screen since it appears to be just one or more spaces. Since this character is essential when using the program **make** (see Section 7) it is important to know whether this character is really in a file. There is one easy way to find ⟨TAB⟩ — just search for it by typing `C-s` ⟨`TAB`⟩. This will find all ⟨TAB⟩'s in the file. There are **emacs** commands for putting in and taking out tabs from a file:

M-x tabify *inside the current region* replace, whenever possible, spaces with ⟨TAB⟩'s.
return
M-x untabify *inside the current region* replace all ⟨TAB⟩'s with with spaces.
return

Occasionally, it may be necessary to put a control character into a file (as opposed to having it be an **emacs** command). This is done by `C-q`.

C-q *char* *char* is put into the file where *char* can be *any* single character, including any control character. For example, `C-q C-c` puts the character **C-c** into the file.

Section 3
Electronic Mail

There are several programs available to read email (electronic mail). These include (in alphabetical order) elm, mutt, and pine. If you have never used mail on a UNIX platform before, we encourage you to use pine since the menus make for a more user-friendly experience. If you are a competent user, however, you might find that mutt is more powerful and flexible, as well as follows RFC standards where others do not. We discourage the use of elm as it does not properly deal with attachments. You can read the on-line manual for any of these by typing for example :

```
man pine
```

This document only discusses the use of **pine**.

3.1 Starting and Stopping pine

Start **pine** by simply typing "pine". The first time you use **pine** you will see a message about creating the subdirectory "/home/username/mail" where pine will store your messages. Any messages that you save or personal mailboxes you create will be located in that folder. At this point, you will be presented with a menu:

```
?     HELP                - Get help using Pine

C     COMPOSE MESSAGE     - Compose and send a message

I     MESSAGE INDEX       - View messages in current folder

L     FOLDER LIST         - Select a folder to view

A     ADDRESS BOOK        - Update Address book

S     SETUP               - Configure Pine Options

Q     QUIT                - Leave the Pine program
```

Things are fairly straight forward from this point on. There are lists at the bottom of each screen that show you what commands are available. If you have questions, you can always refer to the man page. To exit pine, simply press Q.

Section 4
Local Guide To Tex

The programs relevant to using TEX are:

- **tex** - TEX with no format preloaded
- **latex** - TEX with latex format preloaded
- **dvips** - convert TEX output to printable output
- **xdvi** - dvi previewer for X-windows

tex, **dvips** and **xdvi** commands have online manual pages. You can find complete information on these commands by typing "man dvips", for example.
The following steps show how to produce printed output from TEX input:

(1) create a TEX input file using your preferred editor (jed, vi, emacs, etc.)

(2) process the TEX input file using tex:

    tex *filename*.tex

    The tex command creates two output files: *filename*.dvi and *filename*.log. The *filename*.log file contains contains the standard output that can be used to correct errors. The *filename*.dvi file is the file that will be printed.

(3) previewing the dvi file
In order to preview a dvi file you must be on a terminal that supports X-windows such as a Sun Workstation or a Mac running MacX. The following command will display the dvi file:

    xdvi *filename*.dvi

(4) printing the dvi file using dvips:

    dvips *filename*.dvi

    dvips converts the output from device independent format to postscript format. By default, the output is sent directly to the printer. If the default is changed, dvips will create a postscript file called *filename*.ps and the user will need to sent print the file explicitly with the following command:

    lpr *filename*.ps

    dvips can also be used to print selected pages. dvips supports the following options:
    -p *number* - first page to be printed
    -l *number* - last page to be printed
    -n *number* - maximum number of pages to be printed

    For example, the command:

    dvips -p 5 -l 10 *filename*.dvi

    will start printing from page 5 and stop on page 10. The command:

    dvips -p 5 -n 5 *filename*.dvi

    will also start on page 5 and end on page 10. For more detail type **man dvips**.

We try to keep our tex installation up-to-date as possible, if you find anything missing or require additional inputs files please send e-mail to *support*. You may also store your own inputs files in the directory **/home/***username***/texmf/tex/**.

Section 5
Sun Floppy Drive

The commands relevant to using the floppy drive are:
- **fdformat** - format a floppy disk
- **eject** - eject a floppy disk
- **volcheck** - checks for media in a drive

5.1 Formating A Floppy Disk

The **fdformat** command is used to put an MS-DOS filesystem on a disk. This makes it possible to easily transfer files from the Sun to a PC or Macintosh. You can MS-DOS format the disk with low density(720k) or high density(1.44 meg) if you have a high density floppy disk. First, insert the disk in the floppy drive.
To MS-DOS format a low density disk use:

```
fdformat -dl
```

To MS-DOS format a high density disk use:

```
fdformat -d
```

5.2 Mounting A Floppy Disk

The command **volcheck** is used to mount an MS-DOS formatted disk. To use this command insert the floppy disk and type:

```
volcheck
```

This will take a few seconds. When it completes, the floppy disk will be mounted on the directory **/floppy/floppy0**. This command can only be used to mount formatted MS-DOS disks.

5.3 Accessing A Mounted Floppy Disk

Once the floppy disk is mounted on **/floppy/floppy0** it can de accessed just like any other directory. For example, to list the contents of the floppy disk use:

```
ls /floppy/floppy0
```

To copy a file to the floppy disk use:

`cp` *filename* `/floppy/floppy0/`*filename*

And to copy a file from the floppy disk to your directory use:

`cp /floppy/floppy0/`*filename filename*

5.4 Ejecting A Floppy Disk

Use the command **eject** to eject the floppy disk when you are done. The following error will happen if you had typed **cd /pcfs**:

```
/floppy/floppy0:  Device busy
```

Simply type **cd** to return to your **home directory** and type *eject* again.