Contents

1 Introduction **2**

2 Logging On **2**
2.1 Connecting to nadja from the SUN lab . . . 2
2.2 Passwords and Security . . . 3

3 File Management **3**
3.1 Directories . . . 3
3.2 Directory Structure . . . 4
3.3 Moving Through the Directory Tree . . . 5
3.4 Listing Files . . . 5
3.5 Wildcards . . . 6
3.6 File Types . . . 6
3.7 File Archives . . . 6
3.8 File Permissions . . . 7
3.9 Printing . . . 8

4 Editing Files and the vi Editor **9**

5 Remote Sessions: Telnet, SSH and X-Windows **10**
5.1 Telnet . . . 10
5.2 SSH . . . 11
5.3 X-Windows . . . 11

6 FTP **11**

7 Helpful UNIX commands **12**
7.1 Redirection . . . 13
7.1.1 Output Redirection . . . 13
7.1.2 Input Redirection . . . 13
7.1.3 Piping . . . 13
7.2 man . . . 13
7.3 cp . . . 13
7.4 mv . . . 14
7.5 rm . . . 14
7.6 grep . . . 14
7.7 more . . . 14
7.8 head, tail . . . 14
7.9 cat . . . 15
7.10 ps . . . 15
7.11 kill . . . 15

8 Lab Session **16**

1 Introduction

In your studies as a CPS student, you will be required to use a UNIX workstation. UNIX is a very powerful operating system that has been in use for about 30 years. If you are familiar with the MS-DOS operating system used on PC computers, then using UNIX will not pose any substantial problems. However, if most of your computer experience is on Windows or Macintosh based computers, then UNIX can be quite intimidating because all the commands are entered from a `command line`.

This handout is by no means a complete explanation of UNIX, but it will cover everything you need to know in order to complete your coursework. The information presented here is fairly generic and should work for most variations of UNIX (IRIX, SunOS, Linux, BSD, *etc.*).

There are numerous online tools for learning to use UNIX and are available at many levels of ability. To find one that suits your needs, go to an internet search engine and do a search on `UNIX tutorial`. If you need help finding an online site, let me know.

2 Logging On

When you sit down in front of a UNIX terminal, one of the first things you see is the `logon screen`. In order the access the machine, you need to type in your `username` and `password`.

NOTE: Unlike MS-Windows based operating systems, everything in UNIX is `case sensitive`. This means that upper-case characters are different than lower-case ones.

For the SUN workstations in Daily, your username is the same as your email username (*i.e.*, if your email address is `jd1234@brockport.edu` then your username is `jd1234`). Your password is the the same password you use to access your campus email.

`NOTE: Passwords are case sensitive!`

For the CPS department computers, you will be issued a username and password. Use these to log on to the CPS department computers. The main computer you will be using is called `nadja`. This is a 4 processor SGI Onyx 2 computer running IRIX 6.4. IRIX is a variant of UNIX. The formal name of this computer is:

DNS Name: nadja.cps.brockport.edu
IP Address: 137.21.216.50

You may also be given an account on `damien`. This is a Pentium III based computer running RedHat Linux 6.2. Linux is another variant of UNIX. The formal name of this computer is

DNS Name: damien.cps.brockport.edu
IP Address: 137.21.216.2

2.1 Connecting to nadja from the SUN lab

There are 2 ways that you can connect to `nadja` from the SUN lab in Dailey Hall. One of these will significantly slow down the speed of your terminal, so only the second method will be described. When you sit in front of the SUN terminals, you see a box asking for your username along with several pulldown menus below. Pull down the menu that says `Options` and select `Remote Login` then `Enter Host Name`. In the box that pops up, enter

```
nadja.cps.brockport.edu
```

and then click on the `OK` box. You will be presented with the login screen for `nadja`. At this screen, enter the username and password that you have been assigned. If everything works ok, then you should see a desktop with the `Toolchest` in the upper left hand corner of the screen and one open terminal (or command) window. At the bottom of the window, you see

```
nadja >
```

This is called the command line. You interact with the operating system through the command line.

Frequently, you will need to have several terminal windows open. To open another terminal window, go to the `Toolchest` and select `Desktop -> Open UNIX Shell`.

To log off of your session, select any open area of the desktop and press the right mouse button. Select `Log Off` from the menu that pops up.

2.2 Passwords and Security

Computer security is an important topic to be aware of. Here are some guidelines to keep in mind whenever you are logged on.

1) **NEVER** give your password to anyone. On most university computers all of your activity is monitored and you are responsible for anything that is done on your account. If someone gains access to your account, they could engage in any number of illegal activities which could have serious consequences, including revocation of your computer privileges.

2) As soon as you log on for the first time, you should change your password. On `nadja`, you can do this by entering the command `passwd` as the system prompt. The operating system will ask you for your old password and your new desired password. You will then have to re-enter your new password as a check.

3) **Selecting a good password**

   a) Passwords should be at least 6 characters long.

   b) Passwords should not consist of words you can find in a dictionary (*e.g.*, `computer`, `windows`). These are trivial for a hacker to break.

   c) Passwords should not consist of consecutive sequences of letters and/or numbers (*e.g.*, `abc123`, `456789`). These are also trivial for a hacker to break.

   d) Your password should have at least one non-numeric or non-alphabetic character (*e.g.*, `$`, `%`, `^`) and contain a mix up upper and lower case letters.

   e) A good way to come up with a password is to pick a word and replace some of the letters with numbers or symbols that look like letters (*e.g.*, `serious` could become `$eR1ou$`, `anagram` could become `@nAgr@m`).

   f) Another way to obtain a password a good password is to use a mnenomic. Pick a sentence with at least 6 words and use the first letter of each word as your password. Remember to also follow items `d` and `e` above. For example, you could take the sentence `All items should be in a bag` and convert this to `@isB!@b`.

   g) Do not use the passwords in parts d), e) and f).

4) While in Dailey, do not leave your terminal unattended while you are logged on. This would give anyone walking by complete access to your account. It would only take a minute for someone to download a password cracker or root kit onto your account and compromise the security of the entire machine.

3 File Management

3.1 Directories

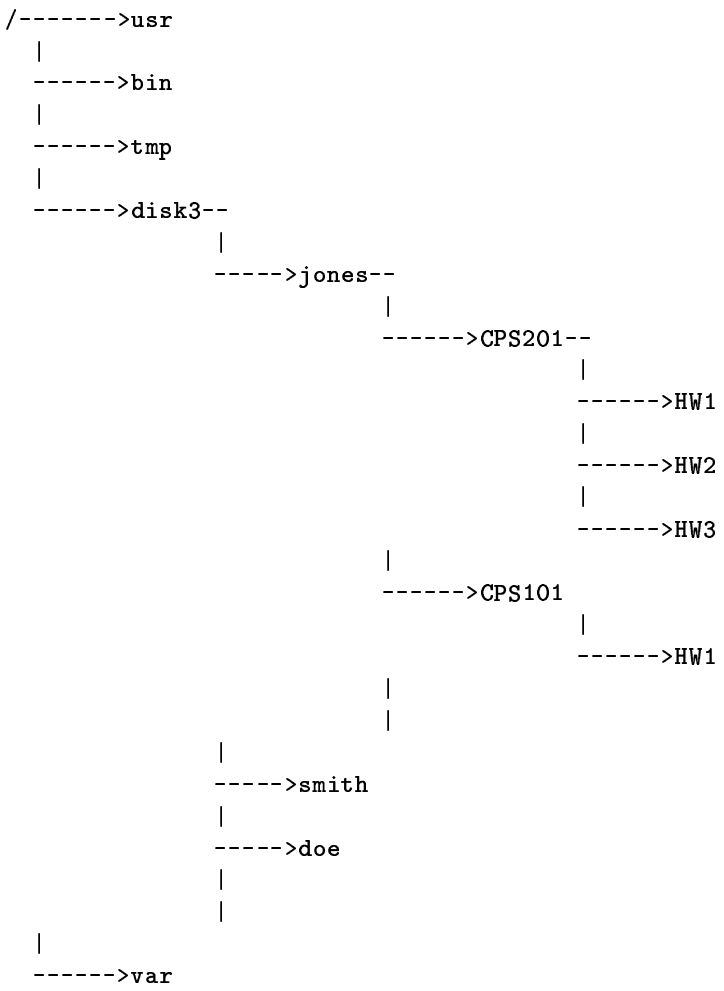
Directories are a way of managing your computer files. You can think of the directory structure as a simple but effective filing system. One important directory is your `HOME` directory. This is the directory you are in whenever you log onto a machine. On `nadja`, if your username is `jones`, your `HOME` directory would look like

```
/disk3/home/jones
```

3.2 Directory Structure

The primary directory for any UNIX machine is the `root` The `path` to this directory is simply `/`. It is possible for a UNIX machine to have many hard drives (for example, `nadja` has hard drives named `/disk2` and `/disk3`) but there is only one top level directory. All other directories are created as `subdirectories` to the root directory.

Directories are arranged in what is called a `rooted tree` structure. Here is an example of a such a structure:



In this example, the user `jones` has created the 2 directories `CPS201` and `CPS101` as branches from their `HOME` directory. Within the `CPS201` directory, additional subdirectories called `HW1`, `HW2` and `HW3` have been created. This provides an easy way to keep the files associated with one homework assignment separate from others.

You should always use directories to organize your files. There are two important reasons for this. The first is that it makes it easier to locate your files. If you have only one directory with thousands of files, locating a specific file can be difficult. The second reason is that this makes it more difficult to accidentally erase all of your files (unfortunately, this is surprisingly easy to do in UNIX).

Here are some useful commands:

`pwd` - displays the directory you are currently in.

`cd dir` - change to the directory `dir`.

`mkdir dir` - creates the directory `dir` as a branch from the current directory.

`rmdir dir` - removes the directory `dir`. The directory must be empty for this command to work properly.

Also, you can always find more information on any command by using the `man` command. For example, entering `man mkdir` would display a screen of information on how to use the `mkdir` command as well as options that are available.

3.3 Moving Through the Directory Tree

In order to better understand moving from one directory to another from the command line, it is important to know the difference from an `absolute path` and a `relative path`. The path is the location of a file in the directory tree. For example, suppose `jones` has a file called `program1.f` in his `HW1` directory. The absolute path of this file is then

```
/disk3/jones/CPS201/HW1/program1.f
```

Specifying the absolute path is always the safest way to be absolutely certain that you are referring to the correct file. Unfortunately, on some computers, an absolute path can be quite long (sometimes consisting of $5-9$ subdirectories) and are inconvenient.

A relative path will get you to the same destination as an absolute path, but is usually shorter and easier to remember. Suppose you are currently in the directory `/disk3/jones/CPS201/HW1` and want to change to the `HW2` directory. You can get there by entering either

```
> cd /disk3/jones/CPS201/HW2
```

or

```
> cd ../HW2
```

The initial portion of the directory path in the latter command is the relative path. It says you want to move 1 directory up in the tree (relative to your current position), then down to the `HW2` directory. Some useful commands for using a relative path are

`cd ..` - change up 1 directory.

`cd ../..` - change up 2 directories.

`cd ../../dir` - changes up 2 directories and then down into the `dir` directory.

`cd ~/` - changes to your `HOME` directory.

`cd $HOME` - changes to your `HOME` directory.

`../HW2/program2.f` - refers to the file `program2.f` in the `../HW2` directory.

3.4 Listing Files

To see what files are in a directory, use the `ls` command. This will show all standard (non-hidden) files and directories in the current directory. There are some important options for this command that are useful.

`ls` - lists standard files and subdirectories in the current directory in alphabetical order.

`ls -l` - the same as the previous command, but the `-l` flag will force the files to be listed in long format. Information such as filename, size, last access time and permissions are displayed.

`ls -a` - lists all files and directories including hidden ones.

`ls -t` - lists all standard files and sorts them in descending order of last access time.

`ls -R` - applies the `ls` command recursivly (*i.e.*, it lists all the files in the current directory and all subdirectory branches of the current directory.

You can mix these flags in any way you want, (*e.g.*, `ls -lt`, `ls -laR`).

Hidden entries special entries that always begin with a . (period). These files are usually configuration files for specific programs.

3.5 Wildcards

Wildcards are used to refer to a group of files that share common properties or similar names. For example, by convention, FORTRAN source files end with a `.f` or `.f90` extension. C source files end with `.c`, postscript files end with `.ps` and text files end with `.txt` or `.text`. Wildcards are an easy way to refer to, say, all postscript files or all postscript files containing a certain sequence of characters. The symbols for wildcards are

`?` - single character wildcard.

`*` - multi character wildcard.

These are best explained by examples.

`*.f90` - refers to any file that ends with `.f90`

`s*.f90` - refers to any file that starts with `s` and ends with `.f90`.

`s?.f90` - refers to any file that starts with `s` followed by any single character in the second place and ends with `.f90`. Note that in this example, the files `s1.f90`, `s5.f90`, and `sr.f90` are all part of the wildcard specification, but `s56.f90` and `sabc.f90`, are not because they have more than one character between the `s` and the . (period).

`*tn*.*` - refers to any file that contains the characters `tn` anywhere before the . (period).

`*.?` - refers to any file that has a single character after the . (period).

3.6 File Types

Ordinary files (not directories) are divided into 2 types: `ascii` and `binary`.

`ascii` files are files that you can open in a text editor (see Section 4) and see something meaningful (*i.e.*, not sequences of meaningless characters). Some common types of `ascii` files are: text files, postscript files, source code for programs, MATLAB source files, and email messages.

`binary` files store information in a very specific manner and are frequently only meaningful to the application that created them. Common types of `binary` files are: PDF files, executable programs, nearly all files created by MS-Windows applications (*e.g.*, `.doc`, `.xls` and `.ppt` files), archive files (*e.g.*, `.zip`, `.gz`, `.tar.gz` and `.Z` files). You should `NEVER` try to print a binary file.

3.7 File Archives

File archiving is a way to distribute a large number of files of various types. It is also a way to compress (and thus reduce) the amount of disk space that files consume. On PC computers, the most popular archive is the `.zip` format.

The main archiving tool on UNIX computers is the `.tar` (for Tape ARchive) format and the most common compression formats are the `.gz` and `.Z` formats.

The compression tools are easy to use. To compress a file using the `.gz` format, issue the command

```
> gzip file
```

By default, the archived file will have the name `file.gz`. You can change this by using the command

```
> gzip file1 file2.gz
```

To uncompress a `.gz` file, use the command

```
> gunzip file.gz
```

If the file format is `.Z`, you can use the same sequence of commands as above, but replace the `gzip` and `gunzip` commands with `compress` and `uncompress` respectively.

The `tar` command is most often used to place a directory (and all files and subdirectories) into a single file. It is also possible to be selective about which files go into the archive file. This is best explained by a simple examples.

Suppose have a directory called `PACKAGE` that contains a complete software package having numerous subdirectories and files. To place the entire `PACKAGE` directory into a `.tar` file, issue the following command:

```
> tar cvf - PACKAGE > PACKAGE.tar
```

You could use any name for the `PACKAGE.tar` portion of the command, but it should have the `.tar` extension and using the name of the directory as the name of the file helps you to remember what the archive contains.

The `tar` command is also used to extract files from a `.tar` archive. Suppose you have received a file called `NEWPACKAGE.tar`. To unarchive this, issue the command

```
> tar xvf NEWPACKAGE.tar
```

This will unpack the contents of the archive (usually several subdirectories and files, but not always) into some directory. Normally, the directory name is the same as the file archive (`NEWPACKAGE` in this example), but not always. If you are unsure of what directory the archive will go into, you can issue the command `ls -lt`. This will list files in the current directory according to the time and date they were created. The files you just unarchived should be near the top.

3.8 File Permissions

File permissions are perhaps the most confusing issue for new UNIX users. UNIX allows you to control who has access to your files. Suppose you issue an `ls -l` command in your current directory and see the following entries:

```
drwxrwxr-x    2   little   little       4096 Feb 20 08:28 TEXT
-rwxrw-r--    1   little   little     777356 Apr 23 13:55 mprog
-rw-r--r--    1   little   faculty    101201 Feb  6 15:08 mprog.f
(1)           (2) (3)      (4)        (5)    (6)          (7)
```

Note that this information is ordered in columns. The meaning of these columns are as follows:

1) The 10 digit initial character indicates the file type and permissions.

2) The next digit is the number of files contained in this entry (will be greater than 1 if the entry is actually a directory).

3) This is the username of the person who owns the file. Only the owner of a file (or the SuperUser) can change its permissions.

4) This is the `group` that the file exists in.

5) This is the size of the file in bytes.

6) This is the date and time of the last file access.

7) This is the file name.

Some of the items are self explantory, but the first few need some explanation.

4) Group information. When your UNIX account is set up, you may be placed in one or more `groups`. A group is a method of allowing large numbers of people to have access to certain files. For example, it may not be desirable for regular employees to have access to the same files as management, and similarly, it may not be desirable for management to have access to the files for the board of directors. The SuperUser can set up groups named `employee`, `management`, and `board`. These groups can also be set up in such a way that someone in a high ranking group has access to all the files in the lower groups. To see what groups you are in, type

   ```
   groups
   ```

   at the command line.

1) Permission information

   a) The first of the 10 permission digits indicates the file type. It has a `-` if the entry is an ordinary file and a `d` if the entry is a directory. There are other possibilities here, but these will not be discussed.

   b) The next 3 spaces are the `owner` permissions and indicate what permissions the file owner (column 3) has . This information is (in order) read permission (indicated by an `r`), write permission (`w`) and execute permission (`x`). Read permission means that the file can be opened and examined, but nothing else. Write permission means that the file can be altered. Execute permission indicates that the file can be run as a program. If the appropriate alphabetic character is present, then permission is granted, if the `-` is present, then that specific permission is denied.

   c) The next 3 spaces are the `group` permissions. The digits have the same meanings, but they apply to everyone who is a member of the entry's group list (column 4).

   d) The next 3 spaces are the `other` permissions and apply to everyone else who is not the owner or in the group list.

If the sample directory entries are examined, it can be seen that `TEXT` is a directory, the owner permissions are `rwx`, the group permissions are `rwx` and the other permissions are `r-x`. The file is owned by user `little` and is also in group `little`. Similarly, the entry `mprog.f` is a regular file, the owner permissions are `rw-`, the group permissions are `r--` and the other permissions are `r--`. The file owner is `little` and is in the `faculty` group.

You can alter the permissions of a file using the `chmod` command. The easiest way for new users to change the permissions is using the flags `u` for the file owner, `g` for the group access and `o` for the others. You use these flags along with the `r`, `w` and `x` flags and the + and - operators (respectively) to grant or remove permissions. Here are some examples:

* **chmod ugo+rwx file** - grant everyone read, write and execute permission to `file`.
* **chmod o-wx file** - remove the write and execute permissions to `file` for the `other` users.
* **chmod g+r-w file** - grant read access and remove write access to `file` for the group.

3.9 Printing

Printing procedures can vary greatly depending on the system you are printing from, hence, only the procedures for the CPS department computers will be covered. The majority of the time, you will be printing ordinary text files (*i.e.*, program listings) and postscript files.

To print either a text or postscript file from `nadja`, just issue the command

```
> lp file
```

The operating system will be able to distinguish the type of file automatically. The output will appear on the printer in the CPS computer lab. This printer sees heavy use, so you should avoid making numerous copies, particularly of long program listings.

The printing situation is somewhat different on `damien`. To print a postscript file, enter

```
> lpr file.ps
```

To print a text file, enter

```
> mpage -n file | lpr
```

In the above command, `n` is the number of reduces pages you want to appear on each page. Good values are 2 and 4. This is good to use for program listings because it conserves paper and you can see more of your program as one time. The printout will also appear on the CPS department printer.

4 Editing Files and the vi Editor

Editing text files is an important aspect of working in the UNIX environment. Text files are files such as email messages and source code for programming languages. The most useful editor in general situations is probably the Visual Integrator (`vi`) editor. There are many editors out there, but this is the one that is used most often and it is probably the only one that is guaranteed to be available on all UNIX machines. Unfortunately, it takes some getting used to because it does not have a graphical interface.

Knowing how to use a text editor is important. This is because you frequently need to edit files on a computer that you are physically not sitting in front of. This will happen if you are working on `nadja` or some other machine from home.

To start editing a text file with `vi`, enter

```
> vi filename
```

at the command prompt. If the file already exists, it will be opened, otherwise a new file will be created.

When you first start `vi` you are placed in `command` mode. In command mode, you can do things like save the file, exit the editor, jump to a specific location in the file and search for a string of characters. In order to actually enter information, you need to enter `input` mode. To do this, enter one of the commands below:

`i` - begin inserting text before the cursor location.

`a` - begin inserting text after the cursor.

`o` - open a new line below the current one and begin inserting.

`O` - open a new line above the current one and begin inserting.

To stop entering text, press the `ESC` key. Some useful commands while in command mode are:

`<CRTL>f` - move one page forward in the file.

`<CRTL>b` - move one page backward in the file.

`/string` - search forwards through the file for `string`.

`?string` - search backwards through the file for `string`.

`nG` - jump to line number `<n>`.

`G` - jump to end of the file.

`yy` - place (yank) the current line into a temporary buffer.

`nyy` - yank `n` lines starting from the current one into a temporary buffer.

`p` - place the contents of the temporary buffer at the current cursor location. Note: if you enter anything other than cursor movement commands after the `yy` command, the contents of this buffer are lost.

`x` - delete the current character.

`dd` - delete the current line.

`ndd` - delete `n` lines starting from the current one.

`:w` - saves the file.

`:wq` or `:x` - save the file and exit the editor.

`:q` - quit the editor.

`:r file` - insert the contents of `file` at the current cursor location.

`:g/string1/s//string2` - replace every occurrence of `string1` with `string2`. Note: this can be dangerous to use, particularly when writing programs.

5 Remote Sessions: Telnet, SSH and X-Windows

Frequently it is necessary to be using several workstations at once or connect to a computer you are not sitting in front of. This is called a `remote session`. The computer you want to connect to is called the `remote computer` while the one your are connecting from is called the `local computer`. There are numerous ways to do this, but the most common are `telnet`, `ssh` and `XWindows` sessions.

5.1 Telnet

Telnet sessions are the most common. To telnet to another workstation, go to the command line and enter

```
> telnet remote_computer
```

If the connection is successful, you will be asked to enter your username and password on the remote computer. Naturally, you need to have an account on the computer you are trying to access. Here are some tips on using telnet:

* The username and password will likely be different for each computer you have access to.
* Telnet is a pure text program and you do not have access to any graphical capabilities. This is why knowing how to use a text editor is crucial.
* Allowing telnet sessions on a computer running any distribution of linux is not recommended unless you are very knowledgeable about computer security.

Once you are connected, you can now perform any usual UNIX tasks on the data and programs you have on that computer.

5.2 SSH

SSH stands for Secure SHell. Telnet is a convenient way to access a remote computer, but is has some serious security deficiencies. In particular, any data transmission you send or receive while running telnet (including your password) is sent in ordinary text. It is relatively easy for someone running a cracking program to intercept your password.

Secure Shell avoids this issue by encrypting all transmissions using a 128 bit encryption scheme. In order to use SSH to connect to a computer, your local computer must have ssh installed on it and the remote computer must be capable of accepting SSH sessions. To start an SSH session, go to the command line and enter

```
> ssh remote_computer -l remote_username
```

You need the `-l` flag if your remote username is different from your local username.

Once you are connected, an SSH session behaves just like a telnet session.

NOTE: You can telnet or ssh into `nadja`, but ONLY `ssh` connections are allowed into `damien`.

5.3 X-Windows

X-Windows sessions operate much like telnet and ssh sessions except that you have access to the graphical capabilities of the remote computer. Any graphics you create on the remote computer will be displayed on your local terminal. When you connect to `nadja` from the SUN lab in the manner described in Section 2.1, you are in an X-Windows session. Setting up an X-Windows session is easy to do, but requires several steps:

1) On the local computer, enter the command

```
> xhost remote_computer
```

2) Connect to the remote computer using telnet or ssh.

3) After logging into the remote computer enter

```
> echo $SHELL
```

at the command line. This will tell you your shell. If your shell is `csh` or `tcsh` enter

```
> setenv DISPLAY local_computer:0.0
```

if the shell is `sh`, enter

```
> set DISPLAY=local_computer:0.0
> export DISPLAY
```

Your X-Windows session should now be working.

6 FTP

The File Transfer Program (FTP) is a method of transferring files from one computer to another. In UNIX, this process is done from the command line and can be confusing the first few times you use it.

To start the FTP program, enter

```
> ftp remote_computer
```

You will be asked to enter your username and password on the remote computer.

In order to transfer files, you need to know if the files you are sending are binary or ascii files.

Once you are connected to the remote computer, you can issue the following commands:

`cd dir` - change to directory `dir` on the remote computer.

`ls` - list the files in the current directory on the remote computer.

`lcd dir` - change to directory `dir` on the local computer.

`ascii` - set transfer type to ascii files.

`bin` - set transfer type to binary files.

`get file` - transfer `file` from the remote computer to the local computer.

`put file` - transfer `file` from the local computer to the remote computer.

`mget file` - transfer multiple files from the remote computer to the local computer.

`mput file` - transfer multiple files from the local computer to the remote computer. For this (and the previous) command, `file` is usually some kind of wildcard.

`prompt` - toggle prompting for confirmation of each file transferred.

`user` - re-enter your username and password. Useful if you enter an incorrect password when starting the FTP program.

`quit` or `bye` - exit the FTP program

Here are some helpful pointers for using FTP.

1) The usage of the terms `remote` and `local` are relative to the computer you issue the FTP command from and can have the exact opposite meanings from their usage in the remote session descriptions in Section 5.

2) FTP is available in MS Windows based operating systems, both from the command line and in graphical applications such as WS-FTP and Cute FTP. The latter applications are usually straightforward to understand.

3) Depending on how you connect to the campus computers from home, you may not have the ability to use FTP. Most dialup connections will not allow you to use FTP. Check with your Internet Service Provider (ISP) to see if they will allow this. If you have a cable or DSL connection, you can use FTP from home.

4) Use `mput` and `mget` where possible.

5) If you are transferring many files of different types (ascii or binary) or extremely large files, consider putting the files into one directory, creating a `tar.gz` archive of the entire directory and transferring that instead. This will be more reliable and will also greatly reduce the amount of data that has to be transferred.

7 Helpful UNIX commands

This is not an exhaustive of available UNIX commands, but they are probably the most often used.

7.1 Redirection

7.1.1 Output Redirection

Sometimes you want the output from a command to be placed directly into another file instead of being displayed on the screen. There are 2 commands that can be appended to any command that will allow this.

> `> file` - placing this after any command will put the command output into `file`. If `file` already exists, you may get an error (depending on how the particular system is configured).
>
> `>> file` - This is the append operator. Placing this after any command will append the command output at the end of `file`.

Here are some examples:

> `ls -a > temp` - list all the contents of the current directory and put the list into the file `temp`.
>
> `ls -la >> temp` - list all the contents of the current directory in long format and append the output to `temp`.

7.1.2 Input Redirection

Sometimes you want the input to a command to come from another file instead of being entered at the keyboard. There are 2 commands that can be appended to any command that will allow this.

> `< file` - placing this after any command will for the operating system to get any input to a command from `file`. For coursework, this is most frequently used to input data from a file rather than the keyboard.

7.1.3 Piping

Another variant of input and output redirection is to have the output of one command be used as the input to another command. You can do this with the `pipe` operator (`|`). Here are some examples:

> `ps -ef | grep jones` - list all tasks currently running, then search through the output and display only those tasks that reference `jones`. If you replace `jones` with your own username, you get a list of all tasks you currently have running.
>
> `grep -i the * | more` - search through all text files in the current directory for the word `the` (case insensitive) and page through the results one page at a time.

7.2 man

The `man` command can be used to obtain detailed information on any UNIX command.

> `man command` - display information for `command`.
>
> `man -k keyword` - display all commands that have `keyword` in their description. Useful when you are not sure what command you want.

7.3 cp

The `cp` (for `copy`) command is used to create a copy of a file The general syntax of the `cp` command is

```
cp source_file destination_file
```

Some examples:

> `cp file1 file2` - copy the file `file1` to `file2`.

`cp file1 ../DIR` - copy the file `file1` to the directory `DIR`, located one directory up relative to the current path.

`cp file1 ../DIR/file2` - same as above, but copy the file into a file named `file2`.

`cp -R DIR1 DIR2 - copies the directory` \BVDIR1+, and all its subdirectories and files (the `-R` means recursive) into the directory `DIR2`.

7.4 mv

The `mv` (for `mv`) command is just like the copy command except that the source file is erased. This command can also be used to rename a file.

7.5 rm

The `rm` (for `remove`) command is used to erase files and directories. Naturally, this is a dangerous command to use because there is usually no Recyle Bin on workstations. Some examples:

`rm file1` - removes the file `file`.

`rm *.f90` - removes all files ending in `.f90`.

`rm -r DIR` - removes the directory `DIR`, including all files and subdirectories. This command is convient if you need to erase a large number of files, but usr it with caution because once you delete `DIR`, it's gone forever.

7.6 grep

The `grep` command is used to search through text files for certain keywords. This is a very useful command.

`grep IMPLICIT *.f` - find all occurrences the the word `IMPLICIT` in all `*.f` files in the current directory.

`grep -i IMPLICIT *.f` - same as above, but performs a case insensitive search.

`grep "*jon*" *` - find all occurrences in all files in the current directory of any word that contains the character sequence `"jon"`.

`grep -n "*jon*" *` - same as above, but in addition, print out the line number where the match is found.

7.7 more

The `more` command is used to page through a file. This is handy when you want to look at a very large file without actually loading it into memory

`more file` - page through `file` one page at a time. Press `space` bar to go onto the next page. To stop paging at any time, press `ESC`.

7.8 head, tail

The `head` (`tail`) command is used to display the top (bottom) portion of a text file.

`head file` - show the first 10 lines in `file`.

`tail file` - show the last 10 lines in `file`.

`head -20 file` - show the first 20 lines in `file`.

`tail -20 file` - show the last 20 lines in `file`.

`head -10 file > file2` - same as above, but puts the output into the file `file2` instead of displaying on the screen. NOTE: Depending on the particular system, this may destroy `file2` if it already exists.

`head -10 file >> file2` - same as above, but puts the output at the end of file `file2` instead of displaying on the screen.

7.9 cat

The `cat` (for `concatenate`) command echos the contents of a text file to the standard output. This is useful when you want to combine several files into one.

`cat file1` - displays the contents of `file1` on the screen.

`cat file1 file2 > file3` - place the contents of `file1` and `file2` into `file3`. NOTE: Depending on the particular system, this may destroy `file3` if it already exists.

`cat file1 file2 >> file3` - place the contents of `file1` and `file2` at the end of `file3`.

7.10 ps

The `ps` command is used to display the processes currently running.

`ps` - displays all processes being run in the current logon session.

`ps -u jones` - displays all processes currently being run by user `jones`.

`ps -ef` - displays all processes currently running.

7.11 kill

Computer programs do not always perform as you intend. You may sometimes find it necessary to `kill` a process that has strayed. To find out what processes are running, issue the command

```
> ps -u username
```

You will see something like:

```
  PID TTY      TIME CMD
19178 pts/0    0:00 tcsh
19183 pts/0    0:00 ps
17958 ?        0:00 bgicons
17964 ?        0:01 sabgicons
18051 ?        0:00 sabgicons
18071 ?        0:00 bgicons
```

The important columns are the first, third and fourth. The first column identifies each process currently running by a process id (`PID`) number. The third column lists the actual CPU time spend executing this command (this is usually much shorter than the wall-clock time). The last column is the name of the process.

`kill PID` - `kill` the process with process id `PID`.

`kill -9 PID` - a stronger version of the above command.

You should occasionally do this to make sure you have no errant processes running since these consume system resources. If an entry in the time column is excessively large it may indicate a process that has not terminated properly (of course, it may also be a perfectly legitimate task).

8 Lab Session

If you are not sure about a command, you can use the `backspace` key to change your input or `<CTRL> C` to cancel the command.

1) Log onto nadja from the SUN lab workstations using your assigned username and password. Use the `passwd` command to change your password (you should write this down).

2) What is your `HOME` directory?

3) What is your `username` and `password`?

4) Create the following directory tree structure as a branch of your `HOME` directory:

```
$HOME --  CPS201  --
       |           |
       |           --  HW1
       |
       |  DATA
```

5) What is the command to change the the directory `$HOME/CPS201/HW1`? Give both the absolute and relative paths assuming that you are in your `HOME` directory. Change to the directory above.

6) Edit a file called `test.txt` (using `vi`) containing the following text:

```
Now is the time for all good men to come to the aid of
    their country.

The quick, brown fox jumps over the lazy dog.
```

   Save this file. EXTRA CREDIT: What is unique about the second sentence?

7) Copy the file you just created to the `DATA` directory and rename it `test.dat`

8) Change to the `DATA` directory.

9) USE FTP to login to this account on nadja: `username: testing password: $%3aC&R` Change to the `GET` directory. Retrieve the `ascii` file `input.dat` and the `binary` file `prog.exe` Exit the FTP program.

10) Enter the command `./prog.exe` at the prompt. What happens?

11) Enter the command `chmod ux` prog.exe+ at the prompt. What happens if you repeat step 10)? Why?

12) What command will undo the command in 11)?

13) Enter the command `cat input.dat` Based on what you see, what do you think the program `prog.exe` does?

14) Using `vi`, add your name above the text contained in `input.dat`. Print this file to the printer in the CPS lab.

15) Change back to your home directory.

16) Create a duplicate of the `CPS201` directory (including all files and subdirectories) in the directory `CPS201NEW`

17) Use the `tar` and `gzip` commands to archive the directory `CPS201NEW` into a file called `CPS201NEW.tar.gz`

18) Remove the directory `CPS201NEW` and all its files and subdirectories.