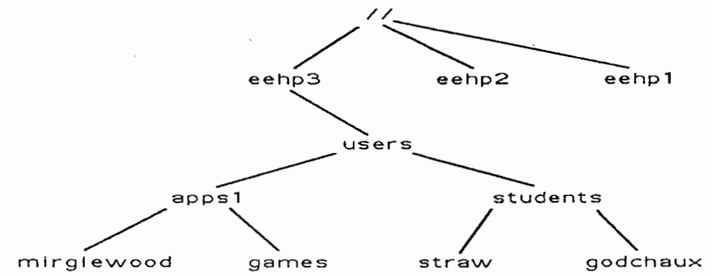
What Resources are available

Above, you were shown some resources that can be applied to x clients, like foreground and background color. There are many different resources that can be specified for an x client. However, there are too many to list here. You might find some resource names for x clients buried in the man pages for those clients. You might also look in manuals in the labs that tell you resource names. You might also want to take a look at the different application default files that are located in /usr/lib/X11/app-defaults. There are several files in this directory that contain quite a few changeable application resources.

The UNIX Filesystem

Introduction

The UNIX filesystem is a hierarchical structure generally organized like a pyramid or tree, as in the following example:



Changing directories (the cd command)

Each of the above headings is a directory, or group of files. The UNIX filesystem is organized into directories, subdirectories, and files, with the "highest" directory being the root.

A directory is identified by its location, given by its pathname. In the above diagram, the absolute pathname on the HP/Apollos for Jack Straw's directory is:

```
//eehp3/users/student/straw
```

This would be Straw's home directory. To see where you are currently located in the file system, type **pwd**. The computer will display your absolute pathname. You need pathnames to change directories, copy files, move files, etc. To change directories, type **cd** and the pathname of where you are going. In the following example, user Straw is leaving his home directory and going down to "minglewood", a subdirectory (see above pyramid):

```
cd /users/applications/minglewood
```

To return to your home directory, type cd without any arguments. To insure that you are home, type **pwd**.

You don't always have to use absolute pathnames to change directories. Some shortcuts are:

```
cd /            takes you to the root directory (on the HPs)
cd /directory   takes you to the directory below the root
cd ..           takes you to the directory immediately above you
cd ../..        takes you up two directories
```

Listing files (the ls command)

To see the contents of a directory, you use the **ls** command. This brings up an alphabetized list of the files in your current working directory. A more complete list can be obtained by applying an option to the ls command. Three common options are **-a**, which lists all files including "invis-

ible" files and subdirectories, **-l**, which displays more information about the contents, and **-g**, which identifies the group that the directory or file belongs to. Using all three options together (**ls -lag**) yields something like the following:

```
drwxr-xr-x  1   jerry   boyz  4096  Sep 9  13:58   ./
drwxr-xr-x  1   jerry   boyz  4096  Mar 29 20:30   ../
-rw-----    1   jerry   boyz  452   Sep 20 14:34   .cshrc
-rw-----    1   jerry   boyz  314   Jan 2  13:47   scr1*
drwx------  1   jerry   boyz  4096  Apr 3  11:33   ochs/
permissions links owner   group size  date of last mod.  filename
```

The first line, named "./", is the current directory. The second line, named ../, is the directory immediately above the current one. The names of directories, like "ochs", always end with a "/" when using the -al option. A filename beginning with . like .cshrc is an invisible file, unless you are using -a.

Wildcards (using the *)

Wildcards are also known as metacharacters, that is, characters that have special meaning to the shell. Wildcards like ? or * serve as shortcuts that allow users to abbreviate filenames when they have a list of similarly named files or they have forgotten a complete filename.

The question mark matches only single characters in a filename, while the asterisk matches any number of characters, including zero characters. For example look at the outcome of the following ls commands.

```
$ ls
gdea
gdead
gdead3
gdead6
gdead12
gdeadh
gdeadgh
agdead
```

```
$ ls gdead?
gdead3
gdead6
gdeadh
```

```
$ ls gdead*
gdead
gdead3
gdead6
gdead12
gdeadh
gdeadgh
```

The "?" matched only those files with a single character after gdead. The * matched files with any number of characters after gdead, but only those beginning with "gdead".

Wildcards can be used with other commands, notably **rm** (remove file). To use this command, type rm and the filename.

CAUTION - Using the asterisk with rm can be very dangerous. You could potentially erase **all** of your files. Before using "rm *", always issue the common "ls *" to know exactly what you are going to delete.

File permissions

Permissions indicate who is allowed to read, write, or execute a file. The first character tells what kind of file you have, such as 'd' for directory. The next nine characters are the actual permissions.

```
- r w x - - x - - x
  owner   group   others
```

Three fields of permission are available on a UNIX file: the file owner, group members, and everyone else. Each of these three fields has three types of permissions. They can read (r), write to (w), or execute (x) the file. Only the owner of the file can set the access permissions for the file. In the above example, the owner can read, write to, and execute the file. Members of the group and all others can execute the file, but cannot read or write to the file.

To change access permissions, use the **chmod** (change mode) command. This allows you to add or subtract permissions, as in the following example:

```
chmod g+rwx scr
```

The 'g' following chmod tells the computer to change permissions for the owner's group. The plus sign will add permission for the specified user class. The rwx refers to the read, write, and execute options. And, "scr" is the file in question. Using a minus sign will remove permissions. The following table explains the shorthand:

- u (user) owner of the file
- g group to which owner belongs
- o (other) all other users
- a (all) takes the place of u, g, o

Differences Between File Systems

There are subtle differences between the HP/Apollo Domain and IBM filesystems. The most notable difference between the two is the concept of a "network root" file system on the HP/Apollo Domain workstations not found on the IBMs.

The HP/Apollo Domains have a network root (//) containing entries with the names of all other HP/Apollo workstations currently on the network. In the this manner the local filesystem (hierarchy) includes not only the files on the local diskdrive, but files and filesystems of all other HP/Apollo workstations on the network. Thus, application programs and home directories can be stored on any machine and accessed by all machines on the network.

Appendix D

Disk Usage

Disk usage

Disk space on all machines is not an infinite resource. All users of the system must do their share to ensure that this resource is available to all other users of the system. There are several things that you can do to keep a tight grip on their usage so that it doesn't get out of hand.

How is disk usage measured?

Disk usage is measured in blocks. Every block equals 1K (or 1024 bytes). So, if a file is 2048 bytes long, it is 2 blocks. However, file space on the IBMs is allocated in 4096 byte chunks. In other words, any file less than 4096 bytes is given a 4 block area.

Thus, if a file is 3548 bytes long, it uses 4 disk blocks. If a file is only 512 bytes, it uses 4 disk blocks. A file using 5000 bytes of space has 8 disk blocks. For example:

```
 ls -al*.c
-rw-r--  1 allender  kaplan  1983  Aug 1 10:02  apply.c
-rw-r-r- 1 allender  kaplan   511  Aug 1 10:02  fund.c
-rw-r--  1 allender  kaplan  2663  Aug 1 10:02  graph.c
-rw-r--  1 allender  kaplan  2414  Aug 1 10:02  input.c
-rw-r-r- 1 allender  kaplan 14947  Aug 1 10:02  mbus-parse.c
-rw-r-r- 1 allender  kaplan 11094  Aug 1 10:02  motif.c
-rw-r-r- 1 allender  kaplan  3602  Aug 1 10:02  nature.c
-rw-r-r- 1 allender  kaplan   367  Aug 1 10:02  utility.c
-rw-r--  1 allender  kaplan 10441  Aug 1 10:02  xdraw.c
total disk space in bytes:  49822 bytes
```

We see that all of these files take up 49822 bytes, but this is the actual size of the files, not the number of disk blocks that are being use. The files apply.c, fund.c, graph.c, nature.c, and utility.c are all using 4 blocks of disk space. The file input.c is using 8 blocks. Motif.c and xdraw.c use 12 blocks and mbus-parse.c uses 16 blocks.

How to check disk usage

There is an easy way to check how much disk space you are taking up. From your home directory, type:

```
du -s .
```

This will give you a sum of how many disk blocks you are taking up. Be sure that you are in your home directory, or you will not get an accurate account of your disk space. Using the command on the example directory above looks like this:

```
du -s .
72 .
```

You can also check the disk usage of individual files:

```
du -s *.c
4   apply.c
4   fund.c
4   graph.c
8   input.c
16  mbus-parse.c
12  motif.c
4   nature.c
4   utility.c
12  xdraw.c
```

The numbers on the left add up to the 72 blocks reported by the original command.

How much space should I be using?

Disk space is different for every person using the system. Some people may be using over 100 blocks, while others may only use 10 blocks. People who are using the workstations for large analysis work will probably be using much more disk space than someone writing machine problems for a computer science class. In general, don't use more disk space than you need. You should only store files in your directory that are pertinent to your class work.

How can I keep my disk usage down?

There are several ways to keep disk usage down. The most obvious way is to remove uneeded files from your directory. Files like "core", and your "*.0" files tend to take up a lot of room. Also, certain programs supported on the workstations tend to generate large data files. Look for files named "*.log" or "*.script". For some jobs, these files can get quite large. If you still need to get rid of some disk space, there are other alternatives.

Each IBM workstation has a floppy drive attached to it. There are some commands that you can use to store files on a floppy disk. You can use the commands **dosread**, **doswrite**, **dosformat**, **and dosdir** to read, write, format, and get a directory listing of an MS-DOS type disk. See the man pages on these commands for more information.

There is a command called **tar** that stands for **T**ape **AR**chive. The tar command takes a set of given files and places those files into what is called a tar file - an archived version of all the files placed there. There are several command line flags that control the operation of tar. We will examine the ones you need.

If you want to create a tar file of some files in your directory, type:

```
tar -cvf <tarfile> <files-to-be-tarred>
```

The c flag tells tar to create a new tar file. The v verifies that the files were tarred correctly. The f flag tells tar what tarfile is being created. Here is an example. Given the directory:

```
-rw-r--  1 allender  kaplan  1983  Aug 1 10:02  apply.c
-rw-r-r- 1 allender  kaplan   511  Aug 1 10:02  fund.c
-rw-r--  1 allender  kaplan  2663  Aug 1 10:02  graph.c
-rw-r--  1 allender  kaplan  2414  Aug 1 10:02  input.c
-rw-r-r- 1 allender  kaplan 14947  Aug 1 10:02  mbus-parse.c
-rw-r-r- 1 allender  kaplan 11094  Aug 1 10:02  motif.c
-rw-r-r- 1 allender  kaplan  3602  Aug 1 10:02  nature.c
-rw-r-r- 1 allender  kaplan   367  Aug 1 10:02  utility.c
-rw-r--  1 allender  kaplan 10441  Aug 1 10:02  xdraw.c
```

If you want to tar these files together, type:

```
tar -cvf Myfile.tar *.c
```

After a time, the prompt will come back and a new tar file called Myfile.tar will have been created:

```
ls -al Myfile.tar
-rw-r-r- 1 allender  staff 61440 Aug 01 13:43  Myfile.tar
```

Notice however, that the size of the tar file "Myfile.tar" is 61440 bytes in size, while the sum of the files (given earlier) is only 49882 bytes. The tar file is always larger than the sum of the size of the files being entered into the tar file. Tar has to store information in the tar file, such as, what files are located in the tar file and how large those files are. So, we have really only succeeded in using more disk space! In addition, tar does not remove the files from your directory that were tarred. Not only do you have a tar file containing the files that were archived, but the original versions of those files still reside on disk. When using tar, be sure that you remove those files that you archived.

We should also verify that tar has indeed stored the files in the archive that we specified. We do this with the -t command line flag to tar:

```
tar -tf Myfile.tar
apply.c             motif.c
fund.c              nature.c
graph.c             utility.c
input.c             xdraw.c
mbus-parse.c
```

You can see that by using the -t flag (in conjunction with the -f flag to specify what tar file to look at) you get a list of the files contained in the archive "Myfile.tar". You would remove the files that were archived with:

```
rm *.c
```

Now, we are only left with the tar file in place of our *.c files.

You will also need to know how to extract files from the archive. This is done with the -x option to tar. So to extract the files that we stored in Myfile.tar, type:

```
tar -xf Myfile.tar
```

This will extract all the files from "Myfile.tar" and place them in the directory where you typed the tar command to extract the files.

Another way to get rid of disk space is to compress your tar file into a smaller file. The format for the compress command is:

```
compress <filename>
```

This takes the specified file and creates a condensed version of that file. The old file is removed and replaced with a new file that has the same name as the old one with the extension .Z added. Watch what happens when we compress the tar file "Myfile.tar" that was created in the last section:

```
compress Myfile.tar
ls -al Myfile.tar*
-rw-r--r--  1  allender  staff  24114  Aug 01 13:43  Myfile.tar.Z
```

You can now see that the size of the tar file has been significantly reduced from 61440 bytes to 24114 bytes. Now, we have saved a little more than half of the disk space that we started out with (49882 bytes). Also notice that the .Z extension was added onto the end of the filename. This indicates that the file is in compressed format.

In order to uncompress an compressed file, type:

```
uncompress <filename>
```

You will always want to use compress and uncompress when storing your tar file locally on the machines. Otherwise, if you use tar without using compress and uncompress, you have only succeeded in using more disk space.

Using TAR to store files on diskette

Each of the IBM workstations has a floppy drive attached to it. You can use the tar command to store your files on diskette, rather than storing them locally on the machines. However, before you tar anything off to a 3 1/2" floppy disk, you must format the disk. Use this command:

```
fdformat -h
```

This formats the disk to high density, about 1.4 megabytes worth of space. Once you have done this, you can use the tar command to place the files onto the floppy. You only need to make a simple change in the command line options to tar to accomplish this. Instead of specifying the tar file to archive files to, we specify the floppy drive as the place to store the files. This is done with the command:

```
tar -cvf/dev/fd0 *.c
```

This will take all the files with the .c extension and tar those files off to a floppy that is in the disk drive. The other commands for verification of tarred files and extraction of files remain the same, except, you now use the /dev/fd0 in place of the filename that was shown in the previous section. The identifier /dev/fdo tells tar that you are using the floppy drive as the location to look at, and not a particular file.

Disk usage abusers

Some people will continually use too much disk space, even after warnings. The system administrators of the labs monitor disk usage of users on the system. Please be aware that if you are using too much space, you will be asked to cut down on your usage. If problems with disk usage persist after the admistrators have already talked with you about the problem, your account may be disabled. It is understood that sometimes you need a large amount of disk space, and as long as it is cleaned up, that is not a problem. But there is no place for those who continually use more that their share of space.

If you need a large amount of temporary file space (24-48 hours) there are temporary file storage locations available on each workstation platform. The IBMs have a file system called "/scratch" and the HP/Apollos have "/local_disk". Be aware that each of these file systems automatically delete all files which are more than 48 hours old.

Appendix E

vi Quick Reference Guide

Introduction

The vi text editor is a mode orientated editor. In a mode orientated editor, the manner in which keystrokes are interpreted depends on the current mode of the editor. In vi, the two main modes are the command mode from which anything typed is interpreted as a command, and the append mode, in which keystrokes are seen as simple text. Note that vi commands are case sensitive.

To invoke vi type:

```
vi filename
```

vi begins in the command mode. The append mode can be started with any of the "Entering Text" commands below. To return to the command mode, hit the <ESC> key.

Below is a list of vi commands:

Scrolling:

^e	Scroll down a line
^y	Scroll up a line
^d	Scroll down a half screen
^u	Scroll up a half screen
^f	Scroll forward a screen
^b	Scroll backward a screen

Cursor Positioning:

j	Down one line
k	Up one line
h	Left one space
l	Right one space
arrow keys	replace **h**, **j**, **k**, and **l** commands
1G	Line 1 of the file
nG	Line n of the file
H	Top of the screen
M	Middle of the screen
L	Last line of the screen
0 (zero)	Beginning of current line
$	End of current line

Appendix E - vi Quick Reference Guide

Entering Text:

a	Append text after cursor
A	Append text before cursor
i	Insert text after cursor
I	Insert text at beginning of current line
o	Open a new line below the current line
O	Open a new line above the current line

Changing Text:

x	Delete character at cursor
dd	Delete current line
ndd	Delete n lines (including current line)
D	Delete remander of line to the right of the cursor
J	Join next line to current line
r<return>	Break line at cursor position

Copying Text:

yy or **Y**	Yank current line to unnamed buffer (does not delete line)
nyy or **nY**	Yank n lines (including current line) to unnamed buffer
p	Put lines from unnamed buffer below current line
P	Put lines from unnamed buffer above current line

Undoing Changes:

u	Undo last command
U	Undo all changes made to current line (before leaving the line)

Searching and Replace:

/pattern	Search for next occurance of pattern
?pattern	Search for preceding occurance of pattern
n	Repeat the last search in the same direction
N	Repeat the last search in the opposite direction
:n,k**s**/old_pattern/new_pattern/**g**	Search for old_pattern and replace with new_pattern in lines n through k
:**%s**/old_pattern/new_pattern/**g**	Global search and replace

Saving Text and Exiting VI:

:w	Write edit buffer to disk and stay in vi
:q	Exit, ignoring changes since last **:w**
:q!	Emphatic form of quit. Use when **:q** fails
:wq	Write edit buffer to disk and exit vi
ZZ	Same as **:wq**
:n,k**w** file	Write lines n through k into file
:n,k**w >>** file	Append lines n through k to file

Appendix F

UNIX Quick Reference Guide

General Information Commands:

!!	Rerun last command issued
cal	Diplay calander for specified year Form: cal year
chfn	Change your information used by the finger command
date	Display the current date and time
du	Summarize disk usage Options: -s display total disk usage (from ./) -a display disk usage of all files and directories
finger	Display information about who is logged in Form: finger Display information about a user Form: finger user's_login
history	List previous commands
hostname	Display name of current host
man	Display manual information for a command Form: man command (see page xx)
whatis	Describe a command Form: whatis command
who	Display information about who is logged in
whoami	Display your login name
who am I	Display more information about your login
users	Display compact list of users on the system

File Manipulation Commands:

cat	Display contents of one or more files Form: cat file#1 file#2 (etc.) Copy file Form: cat file > copyfile Append one file to another file Form: cat added_file >> file Add two or more files to form new file Form: cat first_file second_file > new_file
chmod	Change file permissions (see page xx)
cp	Copy a file to a different filename Form: cp file copyfile
ftp	Start File Transfer Protocol (see page xx)
grep	Display all lines matching a specified pattern Options: -n display line number -i search non-case sensitive Form: grep pattern file

ls	List contents of current directory Options: -l lists more information on files -a list all files, including invisible files begining with a period
more	Display contents one or more files one screenful at a time (press <space> for next screen) Form: more file#1 file#2 (etc.)
mv	Renames a file (be careful not to overwrite existing file) Form: mv old_file new_file Moves a file to specified directory Form: mv file directory
rm	Remove one or more files Form: rm file#1 file#2 (etc.)
sort	Sort lines of a file into alphabetical order Form: sort filename
touch	Creates empty file (or updates time of last access to file if file already exists) Form: touch filename

Directory Manipulation Commands:

cd	Changes working directory Form: cd directory Changes working directory to home directory Form: cd
mkdir	Create a directory Form: mkdir directory
pwd	Display current working directory
rmdir	Remove a directory (directory must be empty) Form: rmdir directory

Abbreviations in Specifying Directories:

.	Your curent working directory
..	Parent (the one above) of your current working directory
~	Your home directory
~user's_login	Home directory of user's_login

Wildcards:

The following are wildcards that can be used for specifying files and directories in most of the commands above.

*	Match any zero or more characters
?	Match any single character
[set]	Match any single character in the set
[a-z]	Match any lower case character
[A-Z]	Match any upper case character
[0-9]	Match any numeric character

Job Control Commands:

&	Run a process in the background Form: command &
bg	Run job in the background Forms: bg (job marked "+" in jobs list is the default) bg process_id# bg %job#
fg	Run job in the forground Forms: fg (job marked "+" in jobs list is the default) fg process_id# fg %job#
jobs	List stopped and background jobs Options: -l display process id numbers as well
kill	Kill process Forms: kill process_id# kill %job# Options: -9 "sure kill" when ordinary kill fails
ps	List all processes belonging to you Options: -a lists processes regardless of position
stop	Stop background job Forms: stop process_id# stop %job#
^z	Stop a foreground job

Redirection:

The shell allows us to redirect standard input to a command and standard output from a command. Forms are:

< file	file is standard input to a command
	Example: mail godchaux < file
	file will be mailed to godchaux
> file	Write standard output from a command to file
	Example: sort file > sorted.file
	file will be sorted and output placed in sorted.file
>> file	Append the standard output from a command onto end of file
	Example: sort file >> big.file
	file will be sorted and appended to end of big.file

Pipes:

The shell also allows the output of one command become the input of another command via a pipe. The form is:

```
command1 | command2
```

The | character tells the shell to connect the standard output of command1 directly into the standard input of command2. You can have as many commands as you wish in a pipeline.

For example:

```
ls -la | more
```

will display the long version of all your files one screenful at a time.

Getting Help

Getting Help

There are several resources available which users can turn to for HELP. Listed below are some of those resources:

Lab operators
Engineering Workstations Users Guide
Quick Reference Guides (available free of charge at labs)
System Reference Manuals (available for check out in each lab)
Software application manual (available for check out in each lab)

In addition to the above resources, users may email questions, problems, or suggestions to our workstation staff. A special mail address (ewsmgr) has been set up on each of the workstations for the purpose of asking questions. To send mail to the "ewsmgr," type:

```
mail ewsmgr
subject: subject of your question
```

Then type your question.