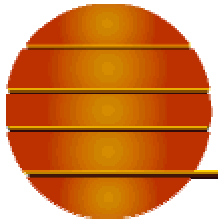
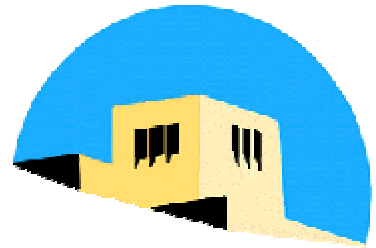
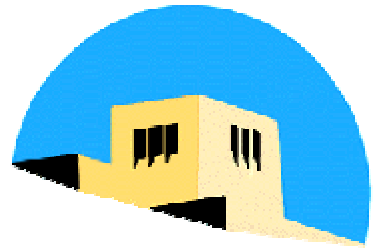




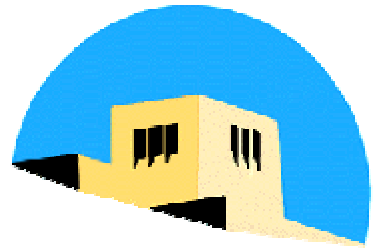
Unix Shell Scripting, Makefiles, X Windows Environment

AHPCC Research Staff



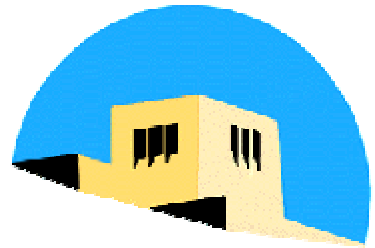
Purpose of this workshop

- To acquaint the user with the basics of the Unix command and programming environment
- Basics and collected useful tricks/HOWTOS



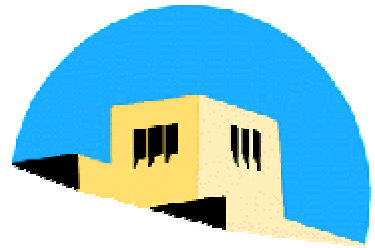
Topics

- Basic Unix Commands
- Regular Expression Syntax
- C shell scripting, Bourne shell scripting
- Make, Makefiles
- Basic X Windows environment configuration



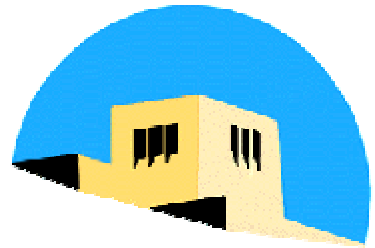
What Is Unix?

- Operating system developed at AT&T in the 1970s
- Portable (moveable) to any platform (not proprietary to a particular hardware vendor)
- Now available as a public domain OS
  - known as Linux
- Reference: Unix is a Four Letter Word...
  - C. C. Taylor, Aug. 1993



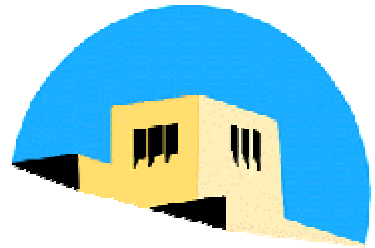
What Is An Operating System?

- A program that:
  - interprets commands you give it
  - provides a file system for your files
  - provides interfaces to peripherals such as printers, disks, tapes, CDs, screens, networks
- Examples of other OSs
  - VMS, DOS, Windows, NT, ...



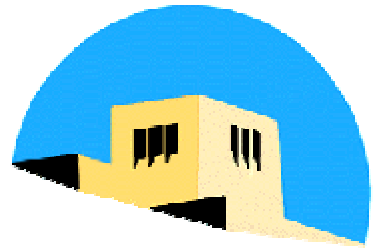
Basics Of Unix

- Commands are case sensitive
  - ls and LS are NOT the same
- The shell is the command line interpreter and there are different shells
  - csh, sh, tcsh, bash, ksh ...
  - they make each Unix look different
  - Most users use csh or tcsh by default



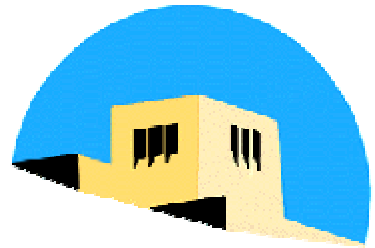
Basics Of Unix Continued

- Command syntax
  - **command [flags] arg1 arg2 ...**
- Examples:
  - **ls -l *.ps**
  - **ls smith**
  - **ls -a**
    - lists all files that begin with the dot character



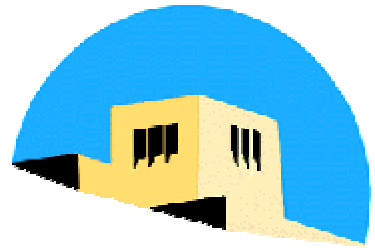
Files And Directories

- Files contain information
  - ASCII characters
  - binary code
  - executable files (binary code)
  - a directory (encoded information about what files are in the directory)
- Directory is a collection of files and other directories



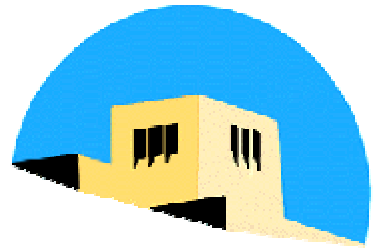
Pathnames

- The entire Unix file system is a series of files, some of which are yours. Devices are files too.
- You get to your files (your desk in the building and a particular drawer in your desk) by specifying a path
- Path names are:
  - /usr/local/bin
  - /home/bsmith or the short form ~bsmith or ~



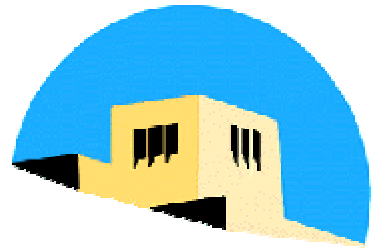
Pathnames Continued

- A pathname is a series of names separated by slashes
- The root file system is /
- Names are a sequences of letters, digits, underscores, dots, ... (other characters but be very careful with some of these)
- Absolute pathnames begin with /



Special Pathnames

- . (a single dot) is the current directory
- .. (double dot) is the directory above the current directory
- ~ is your home directory (csh and tcsh only)
- ~user_name is user name's home directory(csh, tcsh, bash)
- $HOME is the home directory



Relative pathnames

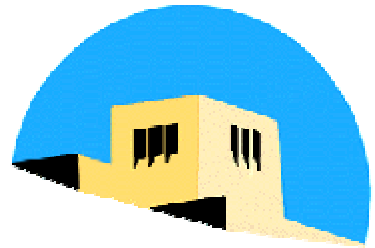
- Let's say you are currently in */home/bsmith* and want to edit a file */home/bsmith/dir/fname.ext* with **vi**. You can use any of:

  vi /home/bsmith/dir/fname.ext

  vi dir/fname.ext

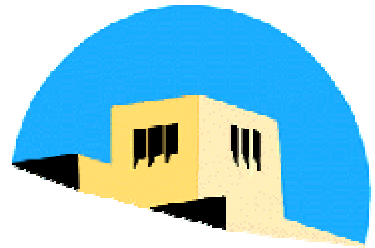
  vi ./dir/fname.ext

  vi ../bsmith/dir/fname.ext



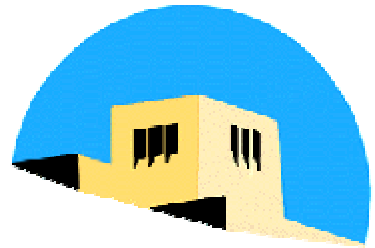
Basic commands

- Copying files
  - **cp [flags] file(s) destination**
  - destination can be a file or directory
  - Analogue: COPY in DOS/Windows and VMS
- Renaming or moving files
  - **mv [flags] file(s) destination**
  - Analogues: RENAME and MOVE in DOS/Windows



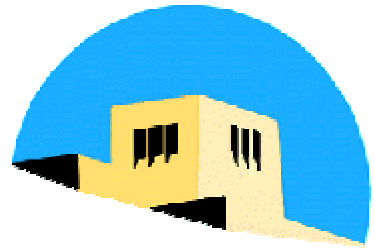
Basic commands continued

- Deleting files [and directories]
  - **rm [flags] file(s)**
  - **rm -r directory**
  - Analogues: DEL, DELTREE in DOS
- Listing files and directories
  - **more file**
  - **ls [flags] [file(s) or directories]**
  - Analogues: MORE and DIR in DOS



Basic Commands - continued

- Changing directories
  - **cd [directory]**
- Creating/deleting directories
  - **mkdir [directory]**
  - **rmdir [directory]**
- Finding out where you are
  - **pwd**



Basic commands - cont.

- File and directory permissions (user, group,world)
  - view with ls -lagF

drwxr-xr-x 1 user group size date/time directory/

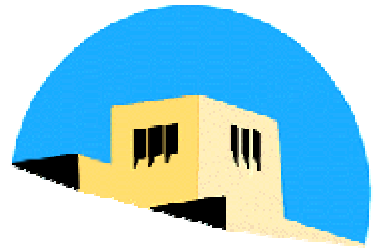
-rwxr-xr-x 1 user group size date/time program*

-rw-r--r-- 1 user group size date/time file

lrwxrwxrwx 1 user group size date/time symbolic_link@

*Note for a directory x means you can **cd** to the directory.*

The number is the number of hard links to a file.



Basic commands - cont.

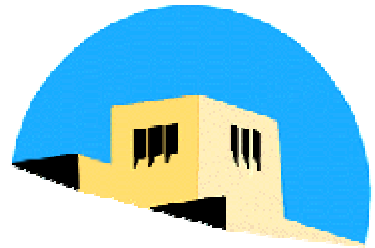
- change with chmod (two forms of command)

chmod u+r,g-w,o-x file/directory

- adds user read permission, and deletes group write and world execute permission.

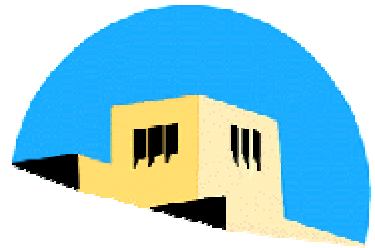
chmod 740 file/directory

- assigns user read, write, execute permission, group read permission, and no world permissions



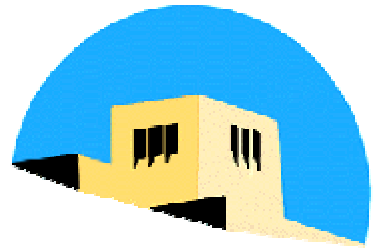
Basic commands - cont.

- Job/process control (* - csh, tcsh, bash)
  - **jobs** (list suspended and background tasks *)
  - **^Z** *or* **^C** (suspend* or terminate current task)
  - **bg [%job]** (run suspended task in backgrnd *)
  - **fg [%job]** (bring task to foreground *)
  - **kill -9 %job [or id]** (terminate task)
  - **command &** (run *command* in background *)
  - **ps** [flags] (show status of processes)



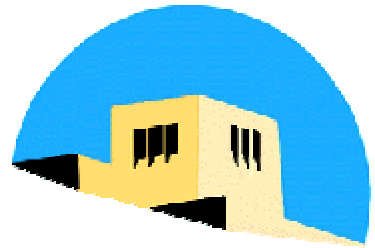
Basic Commands - cont.

- Connecting to remote machines
  - **rlogin host [-l username]**
  - **telnet host**
- Transferring files
  - **ftp host** (over a network)
- Collecting files into a single file
  - **tar cvf archive.tar files_and/or_directories**
  - **tar xvf archive.tar**



Basic File I/O

- Most commands read from standard input and write to standard output, and can be chained together to perform complicated tasks
  - **command < input > output**
  - **command < input >& output**
  - **cmd1 < input | cmd2 | cmd 3 > output**



Changing Password

- Passwords must be changed on a Turing cluster machine.(turingXX.ahpcc.unm.edu)
  - The command is **yppasswd**

    {turing12} 42 % **yppasswd**

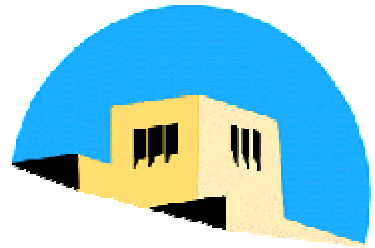
    Changing NIS account information for *username* on taos.

    Please enter old password:

    Changing NIS password for *username* on taos.

    Please enter new password:

    Please retype new password:



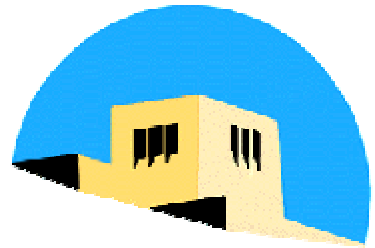
Changing Your Default Shell

- The command **chsh**

server1:~# **chsh**

Changing the login shell for *username*

Enter the new value, or press return for the default Login Shell [/bin/csh]:

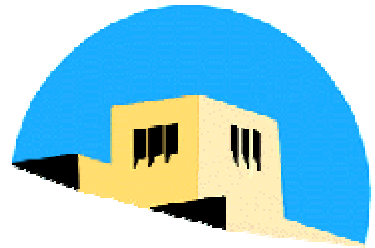


Translate

- Command **tr**
  - substitutes characters from one range of characters for another

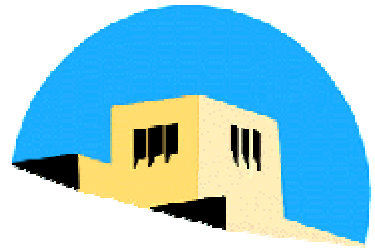
{acoma} 47 % tr ‘[A-Z]’ ‘[a-z]’ <dataInput> dataOutput

- Other options available - see **man tr**



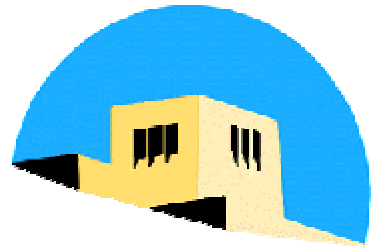
touch

- Changes the access and modification times of each given file to the current time.
- Files that do not exist are created empty.



cat

- concatenates files and print on the standard output
  - % cat fileA fileB fileC > fileD
    - combines files A B and C into file D
  - % cat *filename*
    - shows contents of file without editor



Echo

- Echoes to standard output, what you type.

  % echo What you see_is what you get.

  What you see_is what you get.

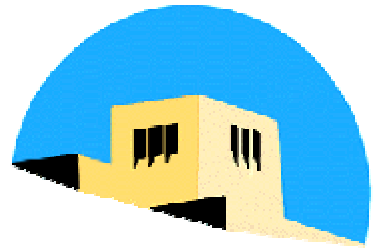
  % echo What-you see.

  What-you see.

  % echo What you see?

  echo: No match.

- Not all characters will echo.

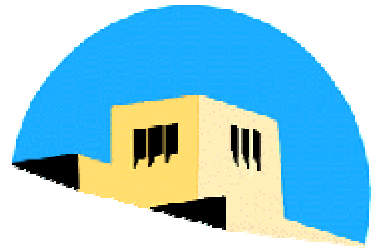


Process Status

- **ps** - Shows current status of processes.
  - Comes in two flavors, so to see your jobs you use different options depending on the system
  - BSD-like UNIXes (Linux, AIX, Solaris)

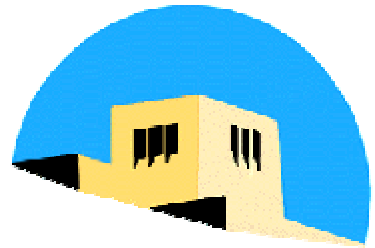
    ps -ux
  - Sys V UNIXes (IRIX, HPUX, Solaris)

    ps -ef



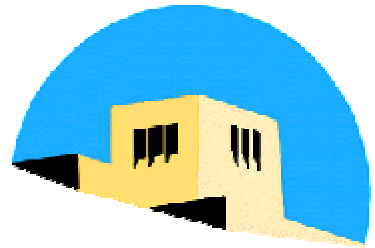
String Editor

- **Sed** edits one line at a time. Default I/O is stdin/stdout
  - Doesn't load a buffer
  - Use redirect to save changes into new file
    - substitution
      - **sed -e 's/Bob/Robert/g' names > Given.Names**
    - deletion
      - **sed -e '/Bob/d' names > temp.names**
  - Useful for modifying filenames in scripts
    - **cp $file `echo $file | sed -e 's/.txt/.f90/'`**



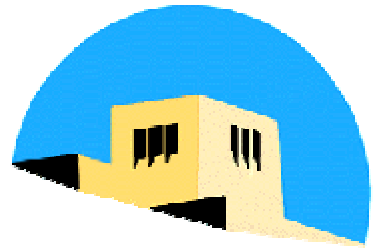
Sort

- sort +2nr -4 +0n -2 X01612_3.OUTPUT -o x.sort
  - This sorts as follows
    - keys are delineated by blanks between fields (default)
    - 1st key, after 2nd field and before 4th = 3rd field
      - n=numeric, r=descending order
    - 2nd key, after 0th field and before 2nd = 1st field
      - n=numeric, ascending by default
    - file to be sorted
    - -o is followed by output file. (without this it would go to the screen unless redirected elsewhere with >)



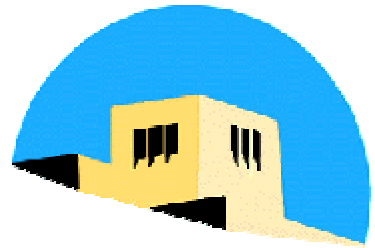
Cut and Paste

- Cut can be used to extract columns based on either character positions or using fields defined by some delimiter (like a : or tab)
- Paste is the inverse of cut. It concatenates files by line instead of sequentially.
- See their man pages for details.

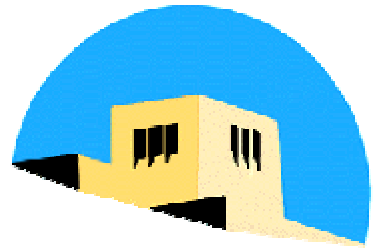


Backing Up/Shipping Files

- Tar (Tape Archive)
  - Creating Archives
    - To File
      - tar cvf file.tar file1 file2 directory1 directory2….
    - To Tape
      - tar cvf /dev/tapedevice file1 file2 directory1 …
        - » rewind device
        - » norewind device



- Tar
  - Listing Archives
    - tar tvf file_or_device
  - Unpacking Archives
    - tar xvf file_or_device
  - Other options allow appending, compression, etc.
- mt command - used to put multiple tar files on a tape.



Compressing Files

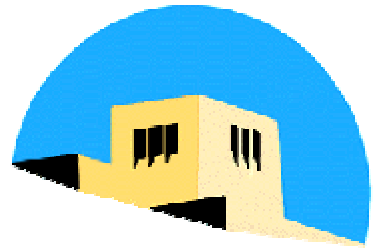
- Compressing single files
  - gzip [-9] file1 file2 … fileN
  - file1 -> file1.gz
- Uncompressing files
  - gunzip [-c] file1.gz file2.gz g*.gz
- gzip/gunzip replaces many older Unix compression programs (compress/uncompress, pack/unpack)

-1 = fast compression

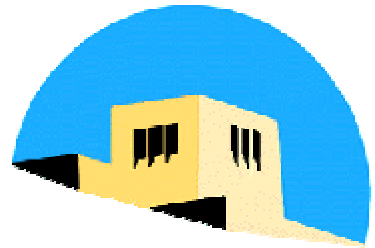
-5 = default

-9 = max compression

-c = keep .gz file

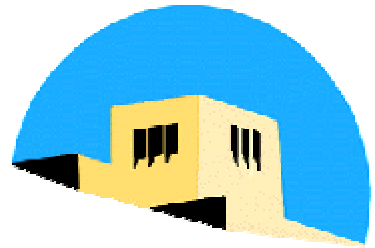


- Mounting removable disk media (floppy, zip)
  - Can’t do as an ordinary user unless root has given permission in /etc/fstab and then only to a specific mount point with specific options.
  - AHPCC desktops have the following mount points with read-write access
  - /dosfloppy, /doszip, /floppy, /zip - the first 2 mount DOS (Win95) formatted media, the last 2 mount Ext2 (Linux) formatted media.



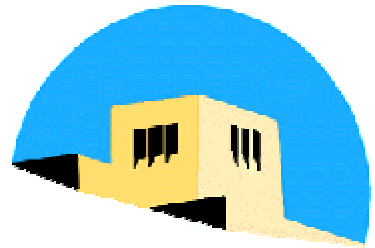
Tracking versions of your files

- Method 1: Use Emacs to number versions
- Method 2 (better): Use revision control. (RCS)
  - lets people work together on files and merge their changes.
  - ci -l filename
    - check file in and creates locked version
    - checked file is filename,v or RCS/filename,v



Version control - cont.

- ci continued...
  - Forces you to document your changes
- co -l [-rn.m] filename
  - checks out a locked version for further editing
  - with -rn.m checks out version n.m for editing
- rlog filename
  - shows history
- rcsdiff -rn.m filename
  - shows differences between current and n.m versions

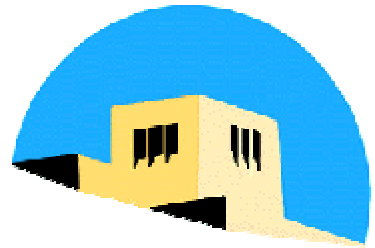


Man and Info

- Man provides information on unix commands - calling syntax, options, usage
- Format:

  man command

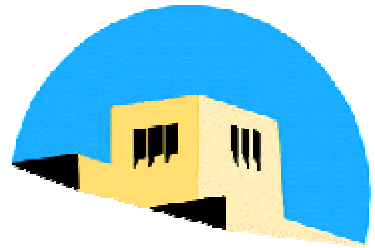
  - becoming obsolete
- Replaced by info



- Format:

  info <command>

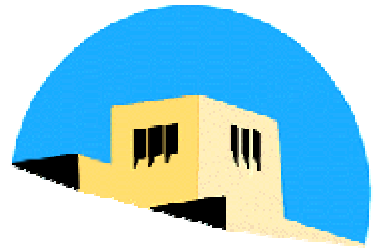
  - provides a somewhat primitive browser and a hyper-linked document that provides information about the named command.
  - Gnu product



man, info exercise

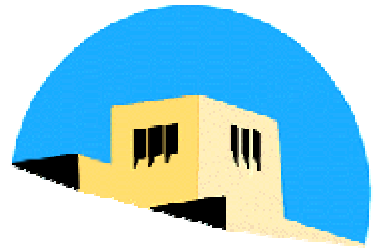
man make

info make



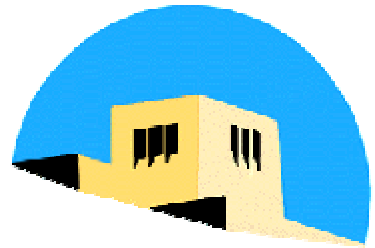
Regular Expression Syntax

- Variants are used in the command line of standard shells.
- It is used in search, search/replace modes in virtually every Unix editor, and text processing program. (Slight variations between programs, e.g. subsets, supersets.)
- Examples: grep, egrep, sed, ed, vi/ex, emacs, perl, awk, tr, more, less



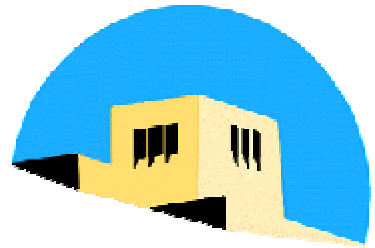
Regular Expression Syntax

- . - matches any single character
- ^ - by itself, matches start of line
- $ - matches end of line
- * - postfix operator meaning match zero or more times (fa* matches f, fa, faa, faaa, …)
- + - postfix operator meaning match one or more times (fa+ matches fa, faa, faaa, …)
- ? - postfix match zero or one instances of the preceeding letter/quantity
- \ - quote special characters



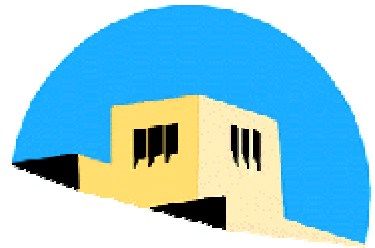
Regular Expression Syntax

- [] matches a character from a set of characters
  - [a-z] matches any single character from the range a-z (assumes ASCII ordering), [adf] matches a,d, or f.
  - [^a-z] matches any single character *not* from the range a-z. [^adf] matches anything except a, d, and f
- \| - alternative matches expression on left or right of \|, e.g. foo\|bar matches foo or bar.



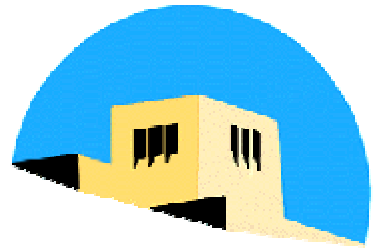
Regular Expression Syntax

- \(… \) - use for grouping, in many ways
  - Encloses a set of \| operations
  - Enclose expressions operated on by *,+, etc.
  - Record a matched substring for later reference in substitutions.
  - The matched expressions are referred to in sequence by \1, \2, \3, …
  - For example,
    - sed -e ‘s/\(Luke\)/\1 Skywalker/g’ < filein > fileout
    - Replaces every instance of Luke by Luke Skywalker



Regular Expression Syntax

- \b matches the empty string at the beginning or end of a word.
  - \btire\b matches tire but not entire.
- \t matches a tab, \n matches a newline, \\ matches a backslash.
- There are a handful of other special sequences that start with \.



File Permissions

```
% ls -l
-rw-r--r--   1 jdoe     gauss          223 Sep 16 15:31 notes
-rwxr-xr-x   1 jdoe     gauss          460 Sep 25 21:26 test
```

```
 uuugggooo
 rwxrwxrwx
```

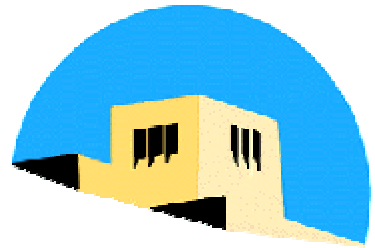
u = user (owner) g = group o = other

r = read w = write (modify and delete) x = execute

The chmod command changes file access permissions.

`% chmod u+x file` **adds execute privileges for file owner.**

`% chmod go-rwx file` **removes all privileges from file for all users except owner.**



Executing files.

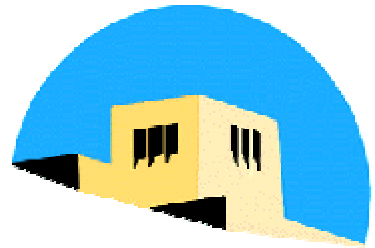
A text or binary file that is executable may be ran by typing its name.

The file must be in a path specified by your path variable, or you must specify the complete path as part of the command.

```
% ls
% /bin/ls
```

A text file that contains commands may be executed with the source command.

```
% source .cshrc
```



Some Useful Commands

echo - displays a line of text

```
% echo this is a test
this is a test
```

more - displays multiple lines of text, usually from a text file.

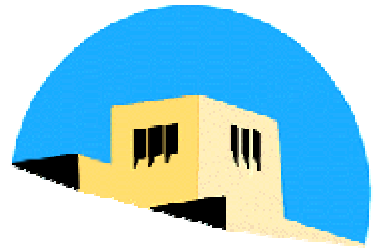
```
% more filename
```

grep - displays lines of text matching a pattern.

```
% grep filename pattern
```

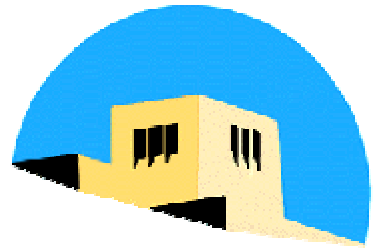
sed - stream editor, edits lines of text.

```
% more filename | sed 's/find/replace/g'
```



Shells

- A shell is a command line parser and interpreter.
- There are many different shells: bash, csh, ksh, tcsh, etc.
- Shells can be run as interactive login shells or script command processors.



Special Characters

Most punctuation symbols have special meanings in UNIX.

- $ indicates start of a variable name.

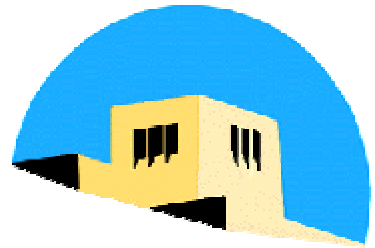
```
% echo prompt
prompt
echo $prompt
%
```

- \ negates the function of the following special character.

```
% echo \$prompt
$prompt
```

- # indicates start of a comment line in a script file.

```
# This is a comment.
```



> Redirects standard output to a file.

```
% echo prompt > filename
```

>& redirects standard output and standard error

| redirects output to another command.

```
% more filename | grep mail
```

`command` executes the argument of a command.

```
% echo hostname
hostname
% echo `hostname`
turing10
```

; separates commands.

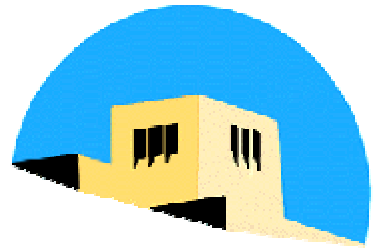
```
% echo one ; echo two
one
two
```

"" - quote a string but allow expansion of variables

```
% set var=foobar
% echo "$var"
foobar
```

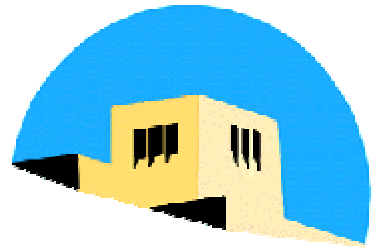
'' - quote a string but don't allow expansion of variables

```
% echo '$var'
$var
```



Variables

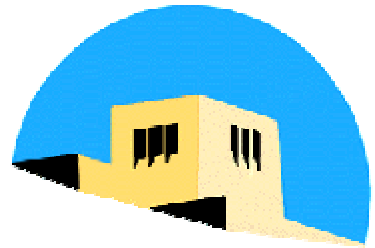
- Variables may be either shell variables or environment variables.
- Shell variables are not maintained when a new shell process is started.
- Environmental variables are maintained when a new shell process is started.
- A dollar sign is used to indicate a variable in most contexts.



Shell Variables

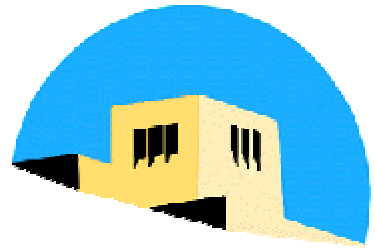
set	*list all defined shell variables*
set name = value	*assign value to name*
set name = (value1 value2 value3)	*assign multiple values to name*
unset name	*remove definition from variable*

```
% echo test
test
% echo $test
test: Undefined variable.
% set test = ( abc def ghi )
% echo $test
abc def ghi
% echo $test[2]
def
```



Environment Variables

setenv, printenv, env	*list all defined environmental variables*
setenv NAME value	*assign value to name*
setenv NAME value1:value2:value3	*assign multiple values to name*
unsetenv NAME	*remove definition from variable*

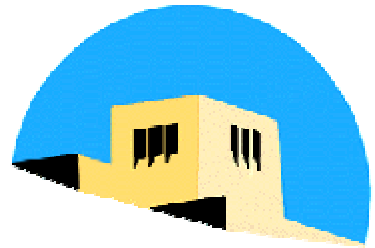


Some Useful Variables

The prompt variable defines the prompt seen in a shell. (C shell)

```
% echo $prompt
%
% set prompt = "] "
]
] set prompt = "`uname -n`> "
turing10> set prompt = “`pwd`> “
/home/jdoe>
```

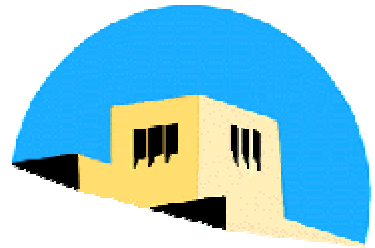
In the Bourne Shell, and variants the prompt variable is **PS1**.



Path variable

The path variable defines the search path for executable commands.

```
% echo $path
/bin /usr/local/bin
% set path = ( $path /home/jdoe/bin )
% echo $path
/bin /usr/local/bin /home/jdoe/bin
```

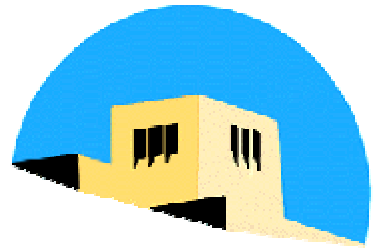


Aliases

Aliases are used to define command substitutions.

`% alias`	*lists all defined aliases*
`% alias name`	*prints the current definition of name*
`% alias name command`	*assigns the given command to name*
`% unalias name`	*removes the definition from name*

```
% alias rm rm -i
% alias change "more \!:1 | sed 's/'\!:2'/'\!:3'/g' > temp ;
   mv -f temp \!:1"
```



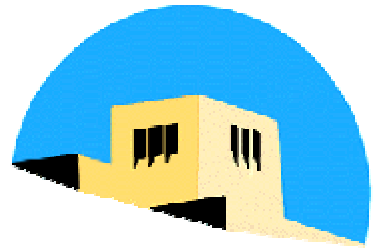
The which Command

Which returns the files which would be executed had its arguments been given as commands.*

Which searches for commands in the directories specified by your path variable.

```
% which ls
/bin/ls
% which set
set: shell built-in command.
% which 2tar
2tar:    aliased to tar -cvf - !:1 > !:1.tar
% which xyz
xyz: Command not found.
```

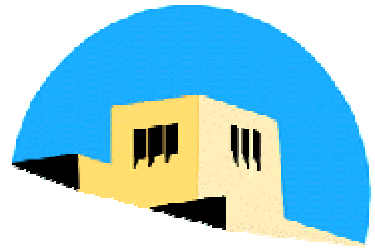
*Taken from which man page (UNIX Reference Manual, 4th Berkeley Distribution)



Customizing your shell environment.

.cshrc	*commands are executed upon starting csh (or tcsh).*
.extension	*sourced by .cshrc file, good place for user commands*
.login	*commands are executed if this is a login shell.*
.profile	*equivalent of .login for Bourne shell and relatives*

/nfs/environment/reset_environment *updates standard dot files.*



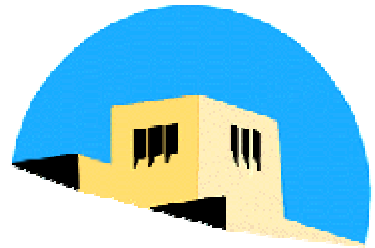
Scripting

Text files that contain commands, or script files, are used to simplify complex procedures, automate repetitive tasks, and in general make using the UNIX/Linux environment easier.

Script files are text files.

Script files often start new shell processes.

Script files often contain commands seldom used interactively at the command prompt.

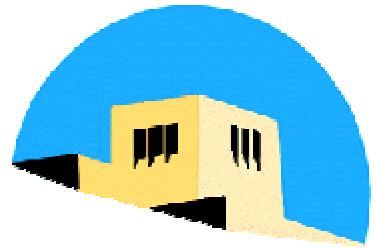


Shell Scripts

A shell script starts a new shell before execution.

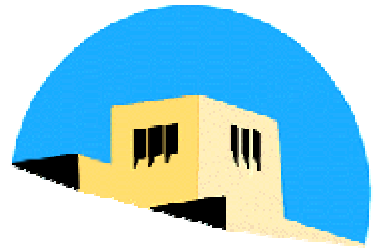
The first line of a shell script specifies the particular shell.

```
#!/bin/csh
```



Flow Control (csh)

```
if(logical expression) command

if(logical expression) then
  commands
elseif(logical expression) then
  commands
else
  commands
endif

exit
```



Logical Expressions

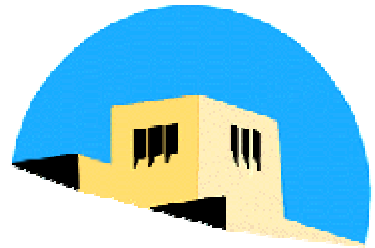
Logical expressions are surrounded by parenthesis.

Logical expressions may be nested.

==	*equal*
!=	*not equal*
||	*or*
&&	*and*

-e file	*tests for file existence*
$?variable	*tests for existence of variable definition*

```
if( ( $TEST == true ) && ( $OSNAME != AIX ) ) echo hello
if( -e test ) rm test
if( $?prompt == 0 ) set prompt = “%”
```



The foreach Command.

foreach name (wordlist)

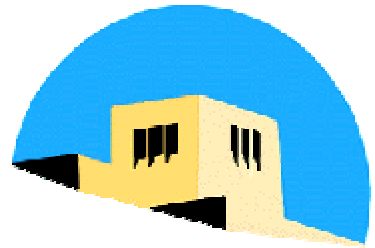
    command

end

The variable **name** is successively set to each member of **wordlist** and the sequence of commands between this command and the matching **end** are executed.*

```
foreach x ( one.f two.f three.f )
   touch $x
end
```

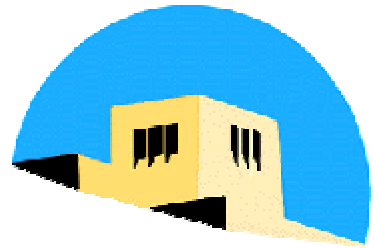
*Taken from csh man page (UNIX Reference Manual, 4th Berkeley Distribution).



Prompting the user for input in a script.

```
#!/bin/csh
echo -n "Number of procs? [4] "
set NNODES = $<
if ($NNODES == '') set NNODES = 4
echo $NNODES
```

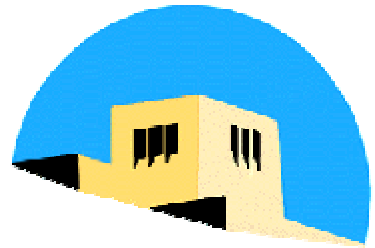
```
% Number of procs? [4] 8
8
```



Passing input arguments on to further commands.

The shell variable **$argv** contains the arguments to the command that started the shell script.

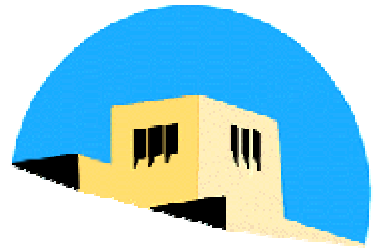
```
#!/bin/csh
# A sample script named xyz
clear
echo $argv
echo $argv[2]

% xyz one two three
one two three
two
```



Sending many lines text or script to a text file.

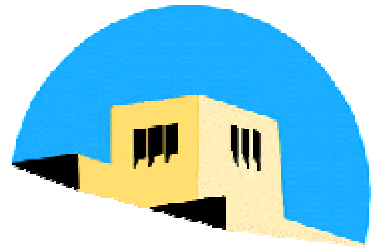
```
#!/bin/csh
# A sample script named xyz
set ONE = 1
set TWO = 2
cat << END_OF_CAT > filename
#this is a test
$ONE
\$TWO
pwd
`pwd`
END_OF_CAT
```

```
% xyz
% more filename
#this is a test
1
$TWO
pwd
/home/jdoe
```



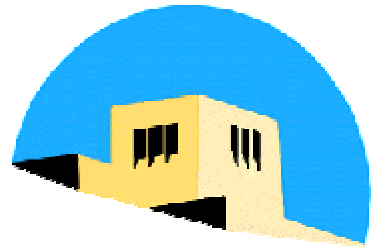
A short example.

```
#!/bin/csh
if (-e ~chem/xmol/xmol.$OSNAME) then
   setenv XUSERFILESEARCHPATH ~chem/xmol/defaults/%N
   xmol.$OSNAME -l $argv &
else
   echo 'No version for' $OSNAME
endif
exit
```



A long example.

http://www.ahpcc.unm.edu/~menlow/gaussian.html



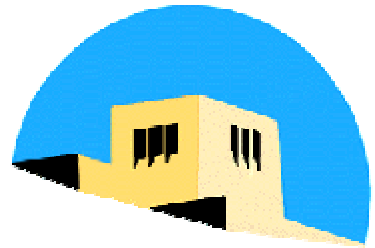
Bourne Shell (sh,bash,ksh)

- If syntax

```
if [ expression ]; then
…
elif [expression]; then
 ...
 else
 …
 fi
```

Example,

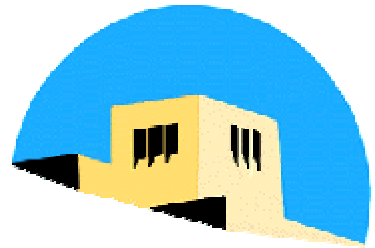
```
if [ -e file ]; then
   rm file
fi
if [! -d directory ]; then
  mkdir directory
fi
```



Bourne Shell (sh,bash,ksh)

- For syntax

```
for var in  list ;  do
    command $var
done
 …
```



Bourne Shell

- Shell/Environment variables

  name=value

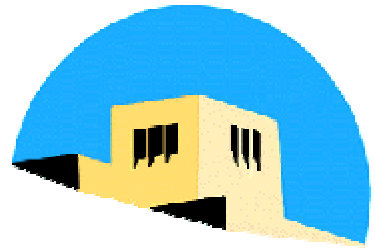
  export name

  export name=value

- Redirecting stdout and stderr

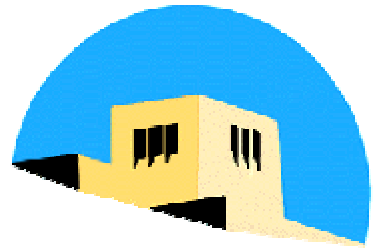
  command > file.stdout 2> file.stderr

  command > file.both 2>&1



Bourne shell

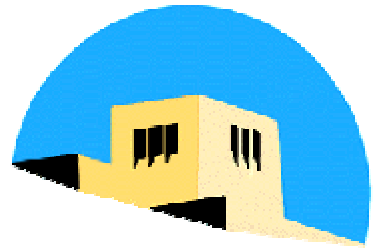
- Running a 2nd command depending on the return status of a previous command. (0=success, greater than 0 = failure)
  - command1 && command2
  - command1 || command2



MAKE

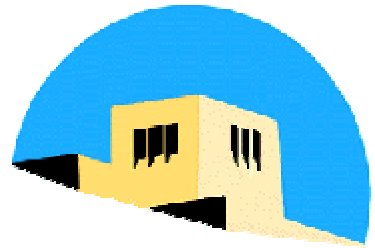
- Reads the makefile and makes the specified target
  - at the unix command line, enter

    make my_exe
  - make will make my_exe according to known rules, following known dependencies
- makes make great for just about anything



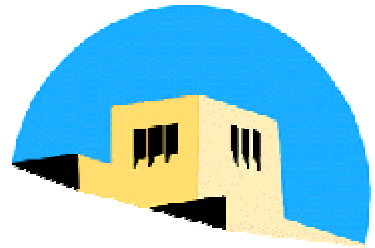
Possibilities...

- executables - f77, f90, hpf, C, C++…
- archived libraries
- documents
- routine tasks
  - source code control
  - batch queue job submission
  - directory maintenance - cleaning, tarring, gzipping...



makefile

- Contains the dependencies and rules to make the specified target(s)
- targets are updated (rebuilt) if their time stamps are older than any of their dependencies
- May use default rules, or at user's discretion, override default rules with specified rules
- May employ variables and included files

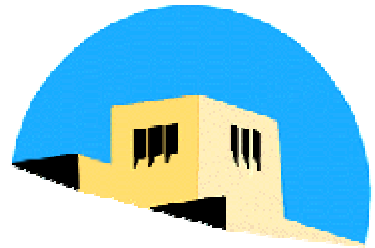


- May have various titles
  - Makefile
  - makefile
  - makefile.sgi

    make -f makefile.sgi pnls
- Examples:

  /home/bennett/workshops/make_lab/…

  http://www.ahpcc.unm.edu/Workshop/Languages/fortran77/fortran77.html#make



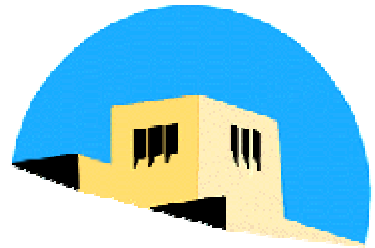
Targets and dependencies

- Target - what you want to make/build
- Dependency - any file that is required to make the target
- Format: in the makefile, signified by

my_exe: my_exe.o /dir/libextra_lib.a

Target

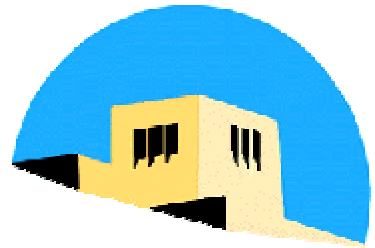
Dependencies



Default target

- The first target encountered in the makefile
- Can override by specifying target name

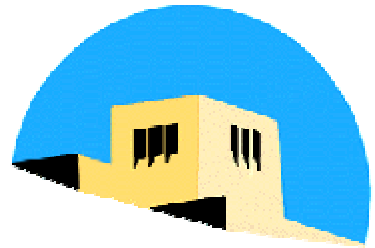
  make mpi_bench



Rules

- Rule - any instruction make is to follow to make the target
- May use UNIX commands
- Format: each line starts with <tab>

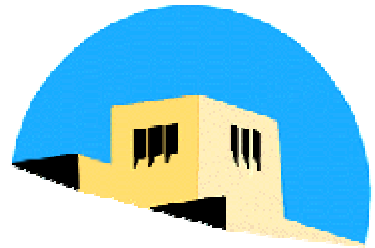
```
f95 -o target.exe target.o -lextra_lib ; cd ..
^tab    ^rules……..^
```



Simple makefile

- To make foo from foo.c, in makefile, 2 lines, link in the C math library

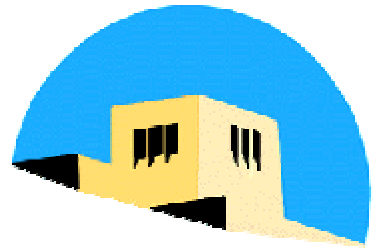
```
foo: foo.c
    cc -o foo foo.c -lm
```



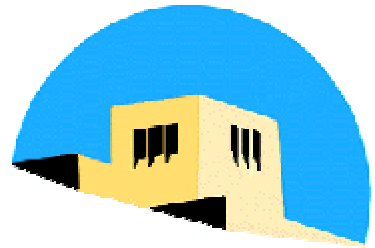
Variables

- A variable is a name that expands to something else
- Format: set variable = list

```
OBJS = main.o sub1.o sub2.o
CC = xlc -qnolm
LIBS = -lm -lpesslp2
CFLAGS = -c -O3 -qhot
```

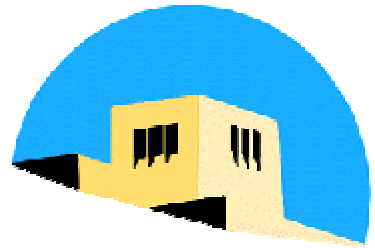


- Simplify the makefile
- Known also as macros
- Some defaults
  - FC = <native fortran compiler>
  - CC = <native C compiler>
  - CFLAGS, FFLAGS, ...



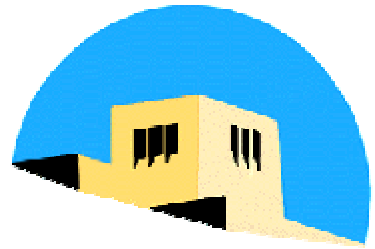
Fancier makefile

```
FC = xlf -qnolm
CFLAGS = -k -O3 -c
LIBS = -lpesslp2
INC = -I/usr/lpp/ppe.poe/include
foo.o: foo.f
	$(FC) $(CFLAGS) foo.f $(INC)
foo: foo.o
	$(FC) -o foo foo.o $(LIBS)
```



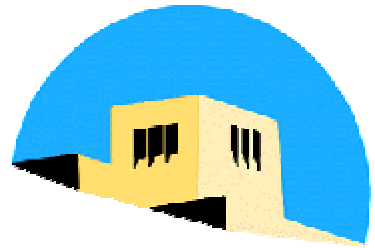
Automatic Variables

- $@ : automatic target variable
- $< : automatic first dependency variable
- $^ : automatic dependency variable for all dependencies with spaces between them
- Used in default rules for building executables from standard sources such as C/C++ code, Fortran code



Even fancier makefile

```
FC = xlf -qnolm
FCFLAGS = -O3 -qhot -qarch=pwr2 -c

foo: foo.o
	$(FC)  -o $@ $<
foo.o: foo.f
	$(FC) $(FCFLAGS) $<
```

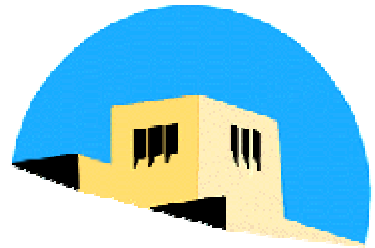


Include

- Used to read in data contained in named files
- Adds great flexibility
- Format:

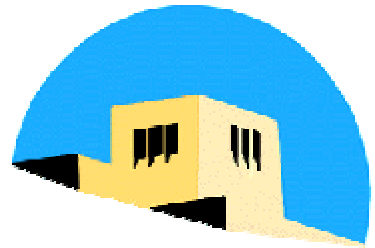
  include <filename>

- Must include before contents needed



Pattern matching rule

- % : for any file that matches the pattern
- Format:
  - %.f matches fortran source name
  - %.c matches C source name
  - %.o matches object file name



Even fancier makefile

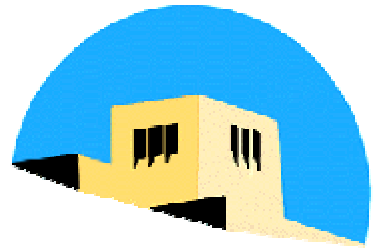
```
include  make_comp
include  make_deps
```

This is a synonym for the original “suffix” rule syntax used for default rules.

.f.o:

$(FC) $(FCOPTS) $< $(INC)

```
%.o: %.f
    $(FC) $(FCOPTS) $< $(INC)
%.mod: %.f
     $(FC) $(FCOPTS) $< $(INC)
$(EXE): $(OBJS)
    $(FC) $(FLOPTS) -o $@ $(OBJS) $(LIBS)
```



Contents of make_comp

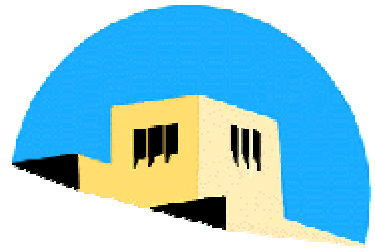
FC = xlf -qnolm

FCOPTS = -c -O3 -qhot -qarch=pwr2

FLOPTS =

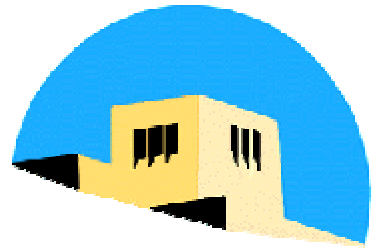
INC = -I/usr/local/mpich/include

LIBS = -L/usr/local/mpich/lib/AIX/ch_eui -lmpi



Contents of make_deps

```
EXE = p2p_bench
OBJS = main.o p2p.o glob.o mpi.o
p2p_bench: $(OBJS)
main.o: main.f mpi.mod
p2p.o: p2p.f mpi.mod
glob.o: glob.f mpi.mod
mpi.o: /usr/local/mpich/include/mpif.f
mpi.mod: /usr/local/mpich/include/mpif.f
```



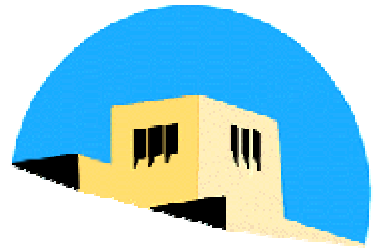
Overriding make

- Use makefile of a different name

  make -f makefile.ibm

- Use a different value for a variable

  make ‘FCOPTS=-O3’



Sample makefiles

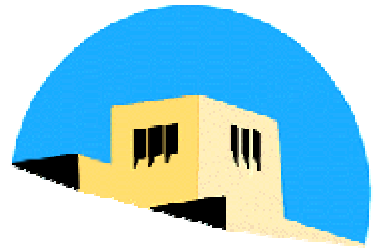
cd /home/bennett/workshops/make_lab/

3 directories containing various working makefiles:

simple/Makefile

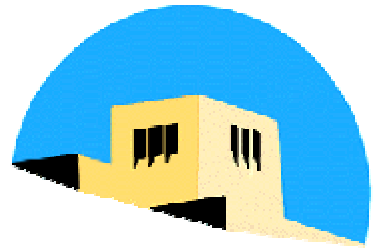
somewhat_fancy/(selected makefiles)

fancy/Makefile,make_(selected endings)



X Windows Configuration

- Files
  - .Xdefaults - settings for X applications
    - Can use different .Xdefault files depending on host!
  - .xsession - startup script for X window session
    - Starts window manager, applications
    - Just a Bourne shell script.
  - .fvwmrc, .fvwm2rc - startup script for fvwm, fvwm2 window managers



Sample portion of .Xdefaults

```
xterm*font:      12x24
xterm*boldfont: 12x24bold
!xterm*font:      courB24
!xterm*boldfont: courBO24
xterm*geometry: 70x28+10-10
xterm*loginShell:        on
xterm*scrollBar:         on
xterm*ttyModes: erase ^H kill ^U
xterm*saveLines:         400
xterm*background: ivory
xterm*foreground: black
xterm*termName: xterm
```

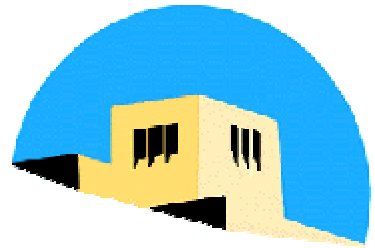
Fonts

Size and location

Erase characters, scroll history

Colors, /usr/lib/X11/rgb.txt

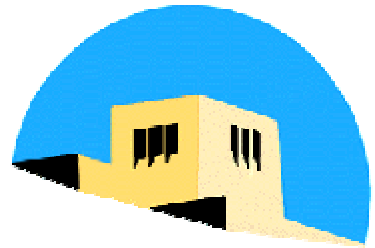
xterm instead of xterm-debian



Sample portions of .xsession

```
#!/bin/sh
DHOST=`basename $DISPLAY :0`
HOSTNAME=`hostname`
...
```

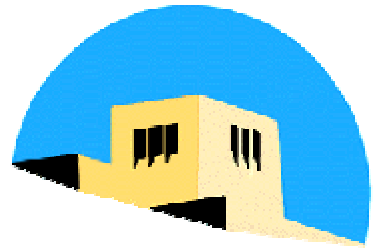
Set some variables that contain the hostname..



Sample portions of .xsession

```
elif test -x /usr/bin/X11/fvwm; then
    if expr $DHOST : "shadow*" ; then
      xrdb -merge $HOME/.Xdefaults.pc
    elif expr $DHOST : "rcde*" ; then
      xrdb -merge $HOME/.Xdefaults.rcde
    elif expr $HOSTNAME : "rcde*" ; then
      xrdb -merge $HOME/.Xdefaults.rcde
    else
      xrdb -merge $HOME/.Xdefaults.linux
    fi
```

Use a different .Xdefaults file depending on whether hostname matches shadow*, rcde*, or a default value. Xrdb -merge sources in the file.



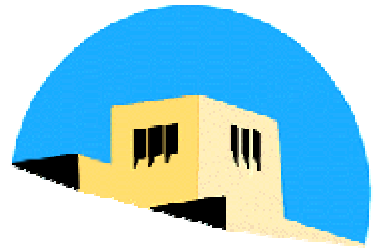
Sample portions of .xsession (continued)

```
exec xsetroot -solid tan &
exec xclock -digital -geometry -0-0 &
exec xbiff -geometry =50x50-0+10 &
if expr $DHOST : "shadow*" ; then
  exec xsetroot -solid tan &
else
  exec xmodmap $HOME/.xkeymap &
fi
exec xterm -geometry =70x7+10+10 -T "FVWM CONSOLE" -
 n CONSOLE -ls  &
exec /usr/bin/X11/fvwm
else
```

Start other applications

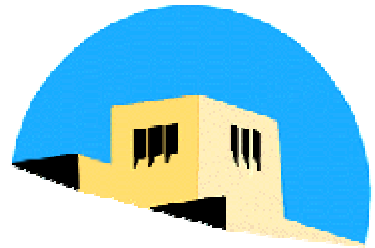
Remap keys when not logged in from certain hosts.

Since fvwm starts last, logs out when fvwm is exited.



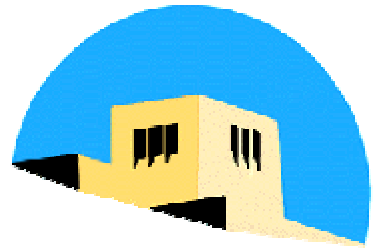
.xkeymap (source with xmodmap)

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```



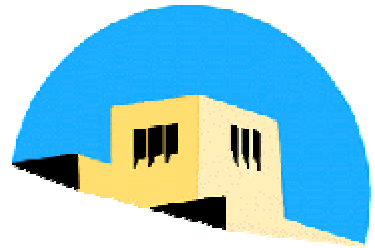
.fvwmrc

```
# this is used for non-selected windows, menus, and the panner
StdForeColor                    Black
StdBackColor                    orange
# this is used for the selected window
HiForeColor                     Black
HiBackColor             purple
PagerBackColor          black
PagerForeColor          orange
MenuForeColor           ghost white
MenuBackColor           purple
# Now the font - there's one for normal use
Font                            -adobe-helvetica-medium-r-*-*-*-180-*-*-*-*-*-*-*
# and one for window title bars
WindowFont                      -adobe-helvetica-bold-r-*-*-*-180-*-*-*-*-*-*-*
```



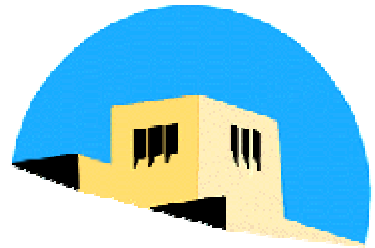
.fvwmrc

```
# Force user to click in window to change focus like Windows
ClickToFocus
#SloppyFocus
# scroll by full pages on the screen edge
# the numbers are the percent of a full screen by which to scroll
# This option replaces NoEdgeScroll and EdgePage from previous
    versions
#EdgeScroll 100000 100000
EdgeScroll 0 0
#EdgeResistance 500 1000
EdgeResistance 250 50
```



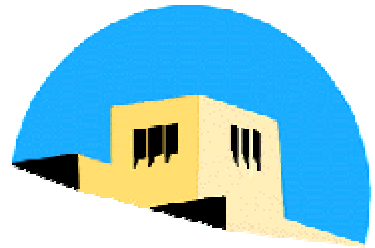
.fvwmrc

```
# Use a random picture the wallpaper
Function "InitFunction"
   Exec "I" exec xv -root /home/user/pictures/*.jpg -random -quit &
   Wait "I" xv
EndFunction
Function "RestartFunction"
   Exec "I" exec xv -root /home/user/pictures/*.jpg -random -quit &
   Wait "I" xv
EndFunction
# Make sure popup window have borders so you can move them!!
DecorateTransients
```



Pull-down menus

- Referred to as pop-ups
- Named
- Contain various entries that may
  - open other pop-ups
  - open other windows
  - execute various tasks



- Defined by

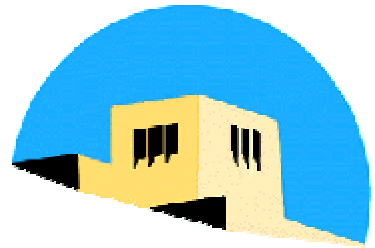
AddToMenu “Popup-name”

+ “New Popup” Title

+ “Old Popup” Popup Prev-Popup

+ “xterm” Exec exec xterm &

+ “mail” Exec exec xterm -e pine &



- Restriction:

  A pop-up that is opened in a different pop-up must be defined prior to the calling pop-up

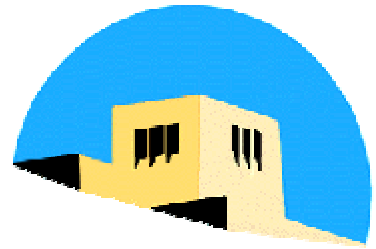
- Example:

  mkdir ~/environment ; cd ~/environment

  cp /home/bennett/workshops/env_lab/.fvwm2rc ./

  view .fvwm2rc

  move down to Popups section



```
AddToMenu  “Quit-Verify” “ Really Quit Fvwm? ” Title
+                        “ Yes, Really Quit “     Quit
+                        “ “                      Nop
+                        “ Restart fvwm2 “        Restart fvwm2
+                        “ Start fvwm95 “         Restart fvwm95
+                        “ Start fvwm “           Restart fvwm
+                        “ Start twm “            Restart twm
+                        “ Start mwm “            Restart mwm
+                        “ Start olwm “           Restart /usr/…
+                        “ “                      Nop
+                        “ No, Don’t Quit “       Nop
```