Environment (esoteric and essential) continued

BASICS OF THE UNIX/LINUX ENVIRONMENT

"people who have trouble with typing commands should not be using a computer."

Response of the Unix community to criticism that Unix ignored the needs of the unsophisticated user.

Environment variables are managed by your shell.

The difference between *environment variables* and regular *shell variables* is that

a shell variable is local to a particular instance of the shell (such as your current shell or a shell script), while environment variables are "inherited" by any program you start, including another shell.

That is, the new process gets its own copy of these variables, which it can read, modify, and pass on in turn to its own children.

In fact, every UNIX process (not just the shell) passes its environment variables to its child processes.

Example environment variable and what it is used for.

PATH

To see the value of the environment variable PATH, echo it to the screen.

PATH
This environment variable tells the shell where to find executable files

```
%echo $PATH
.:/gaia/home/rsmalley:/gaia/home/rsmalley/bin:/gaia/home/
rsmalley/shells:/gaia/home/rsmalley/dem:/gaia/home/rsmalley/
defm:/gaia/home/rsmalley/defm/src:/gaia/home/rsmalley/
visco1d_pollitz/viscoprogs_rs:/gaia/home/rsmalley/gg:/gaia/
home/rsmalley/gg/com:/gaia/home/rsmalley/gg/gamit/bin:/gaia/
home/rsmalley/gg/kf/bin:/gaia/dunedain/d2/gps/bin:/gaia/
smeagol/local/passcal.2006/bin:/gaia/smeagol/local/gmt/
GMT4.2.1/bin:/usr/sbin:/usr/local/teTeX/bin/sparc-sun-
solaris2.8:/gaia/home/rsmalley/bin:/opt/local/sbin:/opt/sfw/
bin:/usr/bin:/usr/ccs/bin:/usr/local/bin:/opt/SUNWspro/SC5.0/
bin:/opt/local/bin:/usr/bin:/usr/dt/bin:/usr/openwin/bin:/
bin:/usr/ucb:/gaia/smeagol/local/bin:/net/gps4/d1/Noah/rbh/
usr/PROGRAMS.330/bin:/gaia/home/rsmalley/X/bin:/gaia/home/
rsmalley/X/com:/gaia/home/rsmalley/record_reading/bin:/gaia/
home/rsmalley/record_reading/scripts
```

```
%echo $PATH
.:/gaia/home/rsmalley:/gaia/home/rsmalley/bin:/gaia/home/
rsmalley/shells:/gaia/home/rsmalley/dem:/gaia/home/rsmalley/
defm:/gaia/home/rsmalley/defm/src:/gaia/home/rsmalley/
visco1d_pollitz/viscoprogs_rs:/gaia/home/rsmalley/gg:/gaia/
home/rsmalley/gg/com:/gaia/home/rsmalley/gg/gamit/bin:/gaia/
home/rsmalley/gg/kf/bin:/gaia/dunedain/d2/gps/bin:/gaia/
smeagol/local/passcal.2006/bin:/gaia/smeagol/local/gmt/
GMT4.2.1/bin:/usr/sbin:/usr/local/teTeX/bin/sparc-sun-
solaris2.8:/gaia/home/rsmalley/bin:/opt/local/sbin:/opt/sfw/
bin:/usr/bin:/usr/ccs/bin:/usr/local/bin:/opt/SUNWspro/SC5.0/
bin:/opt/local/bin:/usr/bin:/usr/dt/bin:/usr/openwin/bin:/
bin:/usr/ucb:/gaia/smeagol/local/bin:/net/gps4/d1/Noah/rbh/
usr/PROGRAMS.330/bin:/gaia/home/rsmalley/X/bin:/gaia/home/
rsmalley/X/com:/gaia/home/rsmalley/record_reading/bin:/gaia/
home/rsmalley/record_reading/scripts
```

The ":" is used to separate each full path name in sh, bash (space for csh, tcsh).

When you run a command (from the terminal or a shell script), your shell looks through each directory in your *PATH* variable , in order, until it finds the first instance of an executable file with the name of the command.

It then runs the command.

```
%echo $PATH
.:/gaia/home/rsmalley:/gaia/home/rsmalley/bin:/gaia/home/rsmalley/shells:/gaia/home/rsmalley/dem:/gaia/home/rsmalley/defm:/gaia/home/rsmalley/defm/src:/gaia/home/rsmalley/visco1d_pollitz/viscoprogs_rs: etc.
```

My path starts with dot ("."). This is convenient but is considered a security weakness.

Next I have a number of my directories where I've written Fortran or C programs and shell scripts.

In the “standard” Unix organization that you will see in most books, one is supposed to put all your executable programs in your “~/bin” directory and all your shell scripts in your “~/scripts” directory.

You will probably not find many people (or systems) that do this anymore.

So how does this work?

If you are working a program to do least squares analysis and decide to call it “ls” what will happen when you enter the command “ls”?

It depends.

What happens depends on your path.

Remember that to Unix, everything outside the kernel (including the shell) is just a file.

Some of these files are executable (programs).

When the shell goes looking through your path for an <u>executable</u> file named "ls", it will run the first one it finds.

If the directory containing your least squares program (executable file), “ls”, is in your path

Before

the directory containing the Unix list command, “ls”, it will run your program and you will not be able (at least simply) to get a listing of your directory!

If the directory containing your least squares program, “ls”, is in your path

after

the directory containing the Unix list command, “ls”, it will run the Unix ls command and you will not be able (at least simply) to run your program!

Example of naming an executable with the same name as a Unix command.

```
smalleys-imac-2:~ smalley$ hello.sh
hello
smalleys-imac-2:~ smalley$ cp hello.sh ls
smalleys-imac-2:~ smalley$ ls
hello
smalleys-imac-2:~ smalley$ rm ls
remove ls? yes
smalleys-imac-2:~ smalley$ ls
!            ESCI7205        Public          hello.sh
Desktop      Movies             bin             unixside
Documents       Music           f1
Downloads       Pictures        f2
```

Force execution of the real ls list command.

```
smalleys-imac-2:~ smalley$ which ls
/bin/ls
smalleys-imac-2:~ smalley$ hello.sh
hello
smalleys-imac-2:~ smalley$ cp hello.sh ls
smalleys-imac-2:~ smalley$ ls
hello
smalleys-imac-2:~ smalley$ /bin/ls
!                Public
Adobe SVG 3.0 Installer Log    Sites
Desktop                bin
Documents              f1
Downloads              f2
ESCI7205          hello.sh
Library                ls
Movies                 sac-101.3-mac_osx.tar.gz
Music             unixside
Pictures
smalleys-imac-2:~ smalley$ rm ls
remove ls? y
smalleys-imac-2:~ smalley$
```

which

Command that shows what the shell finds for the command name.

```
smalleys-imac-2:~ smalley$ which ls
/bin/ls
```

To run a specific executable file - give its full path.

```
smalleys-imac-2:~ smalley$ /bin/ls
!               Public
Adobe SVG 3.0 Installer Log    Sites
Desktop                 bin . . .
```

More examples.
Use all the tricks in specifying paths.

Run from one directory up.

```
smalleys-imac-2:bin smalley$ pwd
/Users/smalley/bin
smalleys-imac-2:bin smalley$ ../hello.sh
Hello
smalleys-imac-2:bin smalley$
```

If the file is in the working directory use the dot.

```
smalleys-imac-2:smalley smalley$ ./hello.sh
hello
smalleys-imac-2:smalley smalley$
```

This behavior is not a “bug”, it is considered to be desirable and an example of the POWER of Unix.

(This is also where the security problem comes in when dot is in your path.)

How you make your path is up to you.

Modifying your environment

Modifying your environment

If you mess up modifying the environment in your current window – you may "break" your current window (shell).

This is generally not a problem on the sun, mac, etc.

The environment is local to that window/ shell.

Just close it and open another window.

How to change/set shell and environment variables.

Use commands set for regular (local) shell variables and setenv for environment (global) variables.

```
set term = xterm
setenv TERM = xterm
```

We already mentioned the difference between regular shell and environment variables.

(you have to know that xterm is something that the shell will understand.)

If you need to deal with this level of Unix, go find a wizard
(Bob Debula, Mitch Withers).

This syntax is also specific to tcsh.

```
set term = xterm
setenv TERM = xterm
```

To do the same thing in bash.

```
term=xterm
TERM=xterm
```

Note that there are NO SPACES on either side of the equals sign here.

How do we tell the difference between a regular shell variable and an environment variable in bash.

(there is really no difference within an instance of a shell).
(set with no parameters lists shell variables, setenv with not parameters lists environment variables, env also lists environment variables.)

When we define it, it is a regular shell variable.

To make it an environment variable (one that is inherited) you export it.

```
term=xterm
TERM=xterm
EXPORT term
EXPORT TERM
```

setenv:

The csh/tchs command to change environment settings.

Can be run on the command line,
from within a local configuration file
(.cshrc or .login),
or in a shell script.

When run without specifying an environment variable, it will print all environment variables to the screen

How to change/set your path.

```
% setenv   PATH    {$PATH}:/gaia/home/rsmalley/scripts
```

This adds the text string (directory)
`'/gaia/home/rsmalley/scripts'`
to the environment variable PATH associated with the active window

When Unix starts, you automatically get a path environment variable (it may be, but probably is not, empty) and this is the best candidate for one you will have to change.

The environment variable is just a text string.

The shell interprets it.

setenv:

```
% setenv   PATH    {$PATH}:/gaia/home/rsmalley/scripts
```

Operationally it adds the the directory

`/gaia/home/rsmalley/scripts`

to the path.

(The braces "{" "}" here are needed, could also use ${PATH}.)

setenv:

```
% setenv    PATH     {$PATH}:/gaia/home/rsmalley/scripts
```

Note PATH is used twice. On the right with the $ it refers to the current value of the environment variable.

On the left it refers to the name of the environment variable that is being set to the string on the right.

setenv:

```
% setenv   PATH    {$PATH}:/gaia/home/rsmalley/scripts
```

Note PATH is used twice. On the right with the $ it refers to the current value of the environment variable.

In this case it will read the current value, append the new information and put everything in a new version of PATH.

If you don't write any of your own programs (or always use the path to the program/file) you will not have to change your path from the default.

(The default path at CERI will give you the path to the tcsh (and other) shell(s), and the paths to the tools such as MATLAB, SAC GMT, and some others.)

Aside ---
How to destroy your input data file.
First - look at file.

```
alpaca.ceri.memphis.edu262:> more flong.dat
1
2
3
4
5
6
7
8
9
10
6
7
8
9
10
alpaca.ceri.memphis.edu263:>
```

Sort it, using the sort command.

```
alpaca.ceri.memphis.edu263:> sort flong.dat
1
10
10
2
3
4
5
6
6
7
7
8
8
9
9
alpaca.ceri.memphis.edu264:>
```

So far OK.

Say we want to save the sorted output to a file. Use redirection.

```
alpaca.ceri.memphis.edu264:> sort flong.dat > flong.dat
flong.dat: File exists.
alpaca.ceri.memphis.edu265:> sort flong.dat >! flong.dat
alpaca.ceri.memphis.edu266:> more flong.dat
alpaca.ceri.memphis.edu267:>
```

We just erased our file!

Unix says we will need an output file, and it has permission to clobber a pre-existing output file - so it does. It then goes looking for the input file - but it just erased it!!

So it sorts nothing (the new empty) and puts it into the output file.

It sees no reason to complain, warn you, etc.

Having no-clobber set prevents this from happening inadvertently (as in our first attempt) as you have to do the >! to get it to clobber the file.

(turning off noclobber returns you to raw Unix.)

Modifying your default environment.

We already saw that you can always change things in your current environment [and that of any new child process] using the <u>setenv</u> command.

But it will get old changing everything to the way you want it each time you log in/open a new window/start a new shell.

And this being Unix, there is a (easy) way to set up your own personal environment.

Modifying your default environment.

The setup of your personal environment (personal changes/preferences for how you want the shell to work for you) in csh and tcsh is stored in the file named

.chsrc

(there is also a file .login, but it is not likely you will have to change it (it get's used when you log in, not each time you start a shell) - so I'll mention it for completeness, but let's ignore it.)

When to make your own environment variables.

Anytime you want a global definition of something.

```
alpaca.ceri.memphis.edu417:> grep rtvel .cshrc
setenv latestrtvel rtvel4_9305_5bv19
setenv LATESTRTVEL $latestrtvel
```

Modifying your default environment variable PATH using the .cshrc file.

We are now doing brain surgery on ourselves.

In a mirror.

This is dangerous.

So--

Make a back up of the current, working .cshrc file before you change it.

Have a second terminal window open in case you mess your file up so completely and break your active window. This way you have another window open to delete the offending file and restore things from the backup file. (Unless you run the command to change it in a window, the environment is static once a window is open.)

You want this window open BEFORE you make the change, as any window opened after the file is saved will use the modified, bad, .cshrc file.

For your path, you will see something like this in your .cshrc file.

```
set path = (. ~ ~/bin ~/shells ~/dem ~/defm ~/defm/src $path )
```

Which uses the set command rather than the setenv command.

The man page for set says
var = value set assigns value to var, where value is one of:
word - A single word (or quoted string.
(wordlist) - A space-separated   list   of words   enclosed in parentheses.

Ex. of changing your path in the current shell using the set command.

First see what your path is (using a script I wrote to put out each entry on a separate line).

```
alpaca.ceri.memphis.edu266:> ExaminePath.sh
.
/gaia/home/rsmalley
/gaia/home/rsmalley/bin
. . .

/gaia/home/rsmalley/record_reading/scripts
alpaca.ceri.memphis.edu267:>
```

Ex. using the command <u>set</u> and the environment variable <u>path</u> to set the environment variable <u>PATH</u> (tcsh).

```
alpaca.ceri.memphis.edu265:> set path = ( $path ~/ESCI7205 )
```

Now look at the environment variable PATH

```
alpaca.ceri.memphis.edu266:> ExaminePath.sh
.
/gaia/home/rsmalley
/gaia/home/rsmalley/bin
. . .

/gaia/home/rsmalley/record_reading/scripts
/gaia/home/rsmalley/ESCI7205
alpaca.ceri.memphis.edu267:>
```

The ESCI7205 entry was not there before.

I've not been able to find documentation on how this works. (I think one is for sh/bash and one for csh/tcsh)

But this is what you will see in both the universal .cshrc (/etc/.cshrc), and if you make changes, in your own .cshrc file.

It has been copied down through the ages.

When you set path, it also changes PATH.
When you setenv PATH, is also changes path.
They seem to track.

.cshrc (csh resource script) configuration file (aka dot file)

```
setenv PATH .:/gaia/home/rsmalley/bin:$PATH
setenv PATH ${PATH}:/gaia/home/rsmalley/record_reading/bin
setenv PATH {$PATH}:/gaia/home/rsmalley/record_reading/scripts
setenv PRINTER 3892
alias cd                    'cd \!*;echo $cwd'
alias home                   "cd ~"
alias del                  'rm -i'
set history=500
set ignoreeof
set savehist=500
set filec
```

Once you have made changes to your .cshrc (and saved them), which is just a file, how do you have them activated in your current window/ shell?

(at this point they will be activated in any new shell/window/login)

You could log out and then log back in (not very efficient, but works), or open a new window and work there.

Use the source command with the .cshrc file as input. (don't need the input redirect "<")

```
alpaca.ceri.memphis.edu151:> source .cshrc
alpaca.ceri.memphis.edu152:>
```

source: executes configuration files

If you change your configuration file, you will need to execute source in all open terminal windows for the changes to take effect. The changes automatically will take effect when new terminal windows/shells are opened.

Say you have edited the .cshrc file.

```
%nedit  ~/.cshrc
%source ~/.cshrc
```

The default .cshrc file that everyone at CERI gets when they login, open a window, start a shell is stored in the file

/etc/.cshrc

After that the shell looks in your home directory for a .cshrc, which is used to expand upon and/or override the CERI values.

MANPATH

Tells the shell where to find the manual pages read using the man command

```
%echo $MANPATH
/gaia/smeagol/local/passcal.2006/man:/gaia:smeagol/local/gmt/
GMT4.2.1/man:/opt/local/man:/ceri/local/man:/usr/dt/man:/usr/
man:/usr/openwin/share/man:/usr/local/man:/opt/SUNWspro/man:/
opt/sfw/man:/usr/local/teTeX/man:/gaia/smeagol/local/man:/opt/
csw/man
```

If you do a man on a command and the shell can't find a manual page (and you are sure the man page exists), this environment variable may not be set correctly.

HOST: environment variable with the name of the machine you are currently logged into.

REMOTEHOST: environment variable with the name of the machine you are sitting in front of, if different (e.g. you are in the class on a PC and have used the program ssh to log into a sun at CERI.).

```
alpaca.ceri.memphis.edu161:> echo $HOST $REMOTEHOST
alpaca.ceri.memphis.edu c-75-66-47-230.hsd1.tn.comcast.net
alpaca.ceri.memphis.edu162:>
```

SSH_CLIENT: the IP (internet protocol) address and port of the HOST machine.

SSH_CONNECTION: the IP addresses and ports of the HOST machine and the REMOTEHOST machine.

```
alpaca.ceri.memphis.edu162:> echo $SSH_CLIENT $SSH_CONNECTION
75.66.47.230 51704 22 75.66.47.230 51704 141.225.157.63 22
alpaca.ceri.memphis.edu163:>
```

If you want to get as much info as you can about the IP addresses. (Can also put in the name and get the address.)

```
alpaca.ceri.memphis.edu169:> nslookup 141.225.157.63
Server:   dns1.memphis.edu
Address:   141.225.253.21

Name:     alpaca.ceri.memphis.edu
Address:   141.225.157.63

alpaca.ceri.memphis.edu170:> nslookup 75.66.47.230
Server:   dns1.memphis.edu
Address:   141.225.253.21

Name:     c-75-66-47-230.hsd1.tn.comcast.net
Address:   75.66.47.230

alpaca.ceri.memphis.edu171:>
```

Aliases

BASICS OF THE UNIX/LINUX ENVIRONMENT

Alias

The alias and unalias commands allow you to rename, or define/undefine "shortcuts" (including mental), for commands.

Their use parallels their name – you are using another name, that is easier to type/ remember, for something.

You can set an alias in your shell interactively (you will only have it locally and in child processes)

or set in your configuration files (.cshrc) so it is available every time you login, start a shell or open a new terminal window (which starts a shell for that terminal window).

Typical Unix think.

When to make/use aliases.

Anytime you find yourself typing the same command over and over, you could make an alias.

Anytime you prefer to type a command "your way".

Typical Unix think.

When to make/use aliases.

Anytime you find yourself mis-typing the same thing over and over, you could make an alias

("mroe" is usually aliased to the more command for example {why learn to type?}. The original, interactive spelling corrector!).

Example aliases taken from .cshrc on CERI system (so you get these automatically).

```
alias   settitlebar    'echo -n "^[]2;$CWD^G"'
alias   cd             'chdir \!* && cwdcmd && settitlebar'
alias   howmuch        'du -sk .'
alias   a              alias
alias   h              'history'
alias   u              unalias
alias   m              more
alias   mroe           more
alias   l              'ls -F'
alias   c              clear
alias   src            source
```

Example aliases taken from my .cshrc file.

```
alias mjday '/gaia/dunedain/d2/gps/oldbin/mjday'
alias home  "cd ~"
alias x     'chmod +x'
alias dir   'ls -lt | more'
alias hp    "lpr -Php_3890 "
alias tek   "lpr -P3904_tek"
alias nb    "lpr -P3892_grad "
alias nbcolor   "lpr -P3892_hpcolor "
alias DEM    "cd $home/dem"
alias ssh_yang 'ssh -l gps yang.soest.hawaii.edu'
alias ftp_jpl 'ftp bodhi.jpl.nasa.gov'
alias matlab_term 'matlab -nodesktop –nosplash'
```

You can find all the aliases that are defined by using the command alias without any arguments.

Dealing with file names with special characters

BASICS OF THE UNIX/LINUX ENVIRONMENT

Say I have a file named "!". (this is probably because I used >! at some time while in bash, but this syntax is for tcsh not bash, so I redirected my output to a file called !)

```
smalleys-imac-2:~ smalley$ rm !
remove !? Y
```

That was easy.

What about a file named "-"

Make a file named "-" with touch command

(use man to see what the touch command does)

```
alpaca.ceri.memphis.edu189:> touch -
alpaca.ceri.memphis.edu190:> ls
-                f2.dat        HW             hw1a.txt       SCRIPTS
f1.dat           f_1_2_3.dat   hw1.txt        NOTES          SRC
```

Try to remove it.

```
alpaca.ceri.memphis.edu191:> rm -
usage: rm [-fiRr] file ...
```

What is the problem? (you tell me.)

We have to let the shell know that the "-" is NOT a switch.

Use the "-" switch all by itself.

```
alpaca.ceri.memphis.edu192:> rm - -
rm: remove - (yes/no)? y
alpaca.ceri.memphis.edu193:>
```

Remember that filenames can have any character but the "/" (used to define the path), so sooner or later you are going to get a file name that will be hard or dangerous to reference.

You will have to be especially careful/ creative if you get a file named "*" as

```
%rm *
```

is disastrous

(and the more privileges you have and the higher up you are in the directory structure, the more disastrous it is.)

File Permissions

BASICS OF THE UNIX/LINUX ENVIRONMENT

Every user on a Unix system has a unique username, and is a member of at least one group (the primary group for that user).

A user can also be a member of one or more other groups.

Only the administrator can create new groups or add/delete group members (one of the shortcomings of the system).

Every file (directories are files) on the system has an owner, and also an associated group.

Every file also has a set of permission flags which specify separate read, write and execute permissions for the

'user' (owner),
'group',
and 'other'
(everyone else with an account on the computer)

Permissions

Read

ability to read the file (r).

Write

ability to write or overwrite the file (w).

Execute

ability to execute or run the file and view directories.

if a directory is not executable, you cannot cd into it or see what is in it at all.

How to view the ownership & permissions of files/direcories (review)

ls -l: lists long format

```
alpaca.ceri.memphis.edu424:> ls -l
total 2201712
-rw-rw-rw- 1 rsmalley user 54847 Mar  7 2009 *CHARGE-2002-107*
-rw-rw-rw- 1 rsmalley user   413 Oct 30 2006 022285A.cmt
-rwxrwxrwx 1 rsmalley user 13092 Aug 13 2007 a.out
Drwxrwxrwx 3 rsmalley user   512 Oct 10 2008 adelitst
Drwxrwxrwx 5 rsmalley user   512 Aug 29 2007 ANT_GMT
```

Permissions

How to view the ownership & permissions of files/direcories (review)

ls -l: lists long format

```
alpaca.ceri.memphis.edu424:> ls -l
total 2201712
-rw-rw-rw- 1 rsmalley user 54847 Mar  7 2009 *CHARGE-2002-107*
-rw-rw-rw- 1 rsmalley user   413 Oct 30 2006 022285A.cmt
-rwxrwxrwx 1 rsmalley user 13092 Aug 13 2007 a.out
Drwxrwxrwx 3 rsmalley user   512 Oct 10 2008 adelitst
Drwxrwxrwx 5 rsmalley user   512 Aug 29 2007 ANT_GMT
```

Owner

How to view the ownership & permissions of files/direcories (review)

<u>ls -l</u>: lists long format

```
alpaca.ceri.memphis.edu424:> ls -l
total 2201712
-rw-rw-rw- 1 rsmalley user 54847 Mar  7 2009 *CHARGE-2002-107*
-rw-rw-rw- 1 rsmalley user   413 Oct 30 2006 022285A.cmt
-rwxrwxrwx 1 rsmalley user 13092 Aug 13 2007 a.out
Drwxrwxrwx 3 rsmalley user   512 Oct 10 2008 adelitst
Drwxrwxrwx 5 rsmalley user   512 Aug 29 2007 ANT_GMT
```

Group

Changing owners and groups.

If you create a file, you are the owner/user.

Mitch and Bob have the system set up to automatically set the group to ‘user’, or all users of the CERI unix system.

Default permissions for CERI files are

rw-r--r--

(numerically 644)

chmod

Command to change file or directory permissions.

```
%chmod ugo+x hello.sh
%ls -lF hello.sh
-rwxr-xr-x   1 rsmalley user      21 Sep 16 08:36 hello.sh*
```

-R flag allows you to set all files to the same permissions within a directory and all subdirectories.

Changing Permissions

you can also use octal values (numbers) to change ownership

644 represents u=rw; go=r
755 represents u=rwx; go=rx

(using this puts you is a special eunuch class)

Connecting remotely

BASICS OF THE UNIX/LINUX ENVIRONMENT

On a mac running OS-X, from a terminal window enter

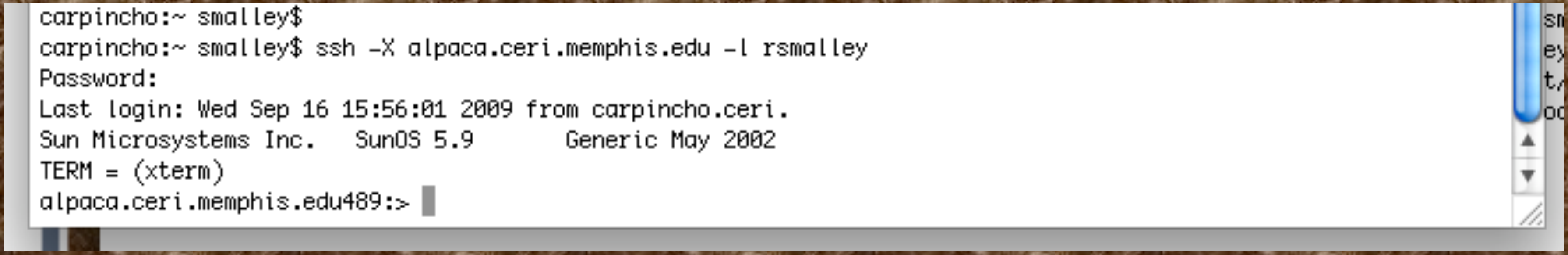
```
ssh –X alpaca.ceri.memphis.edu –l rsmalley
```

The -X flag gives us X-windows graphics capability.

Next is the name of the machine we want to connect to.

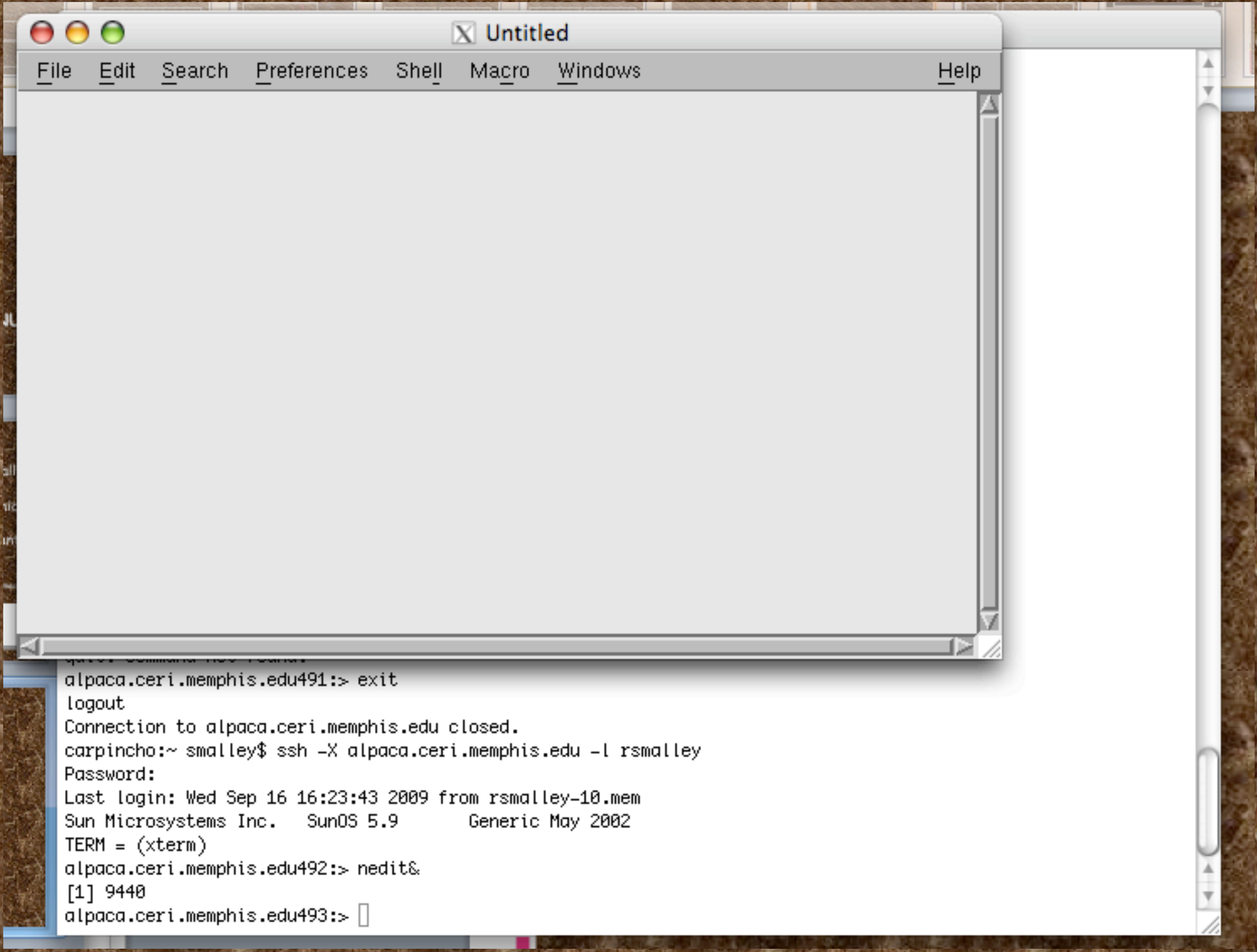
The -l flag passes the username.

(Without this flag it will pass whatever your username is on the mac.)



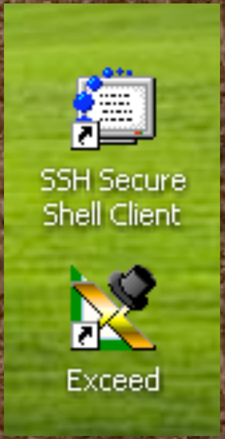
```
carpincho:~ smalley$
carpincho:~ smalley$ ssh -X alpaca.ceri.memphis.edu -l rsmalley
Password:
Last login: Wed Sep 16 15:56:01 2009 from carpincho.ceri.
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
TERM = (xterm)
alpaca.ceri.memphis.edu489:>
```

Try running nedit.
On the mac - we get x graphics automatically



```
alpaca.ceri.memphis.edu491:> exit
logout
Connection to alpaca.ceri.memphis.edu closed.
carpincho:~ smalley$ ssh -X alpaca.ceri.memphis.edu -l rsmalley
Password:
Last login: Wed Sep 16 16:23:43 2009 from rsmalley-10.mem
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
TERM = (xterm)
alpaca.ceri.memphis.edu492:> nedit&
[1] 9440
alpaca.ceri.memphis.edu493:> 
```

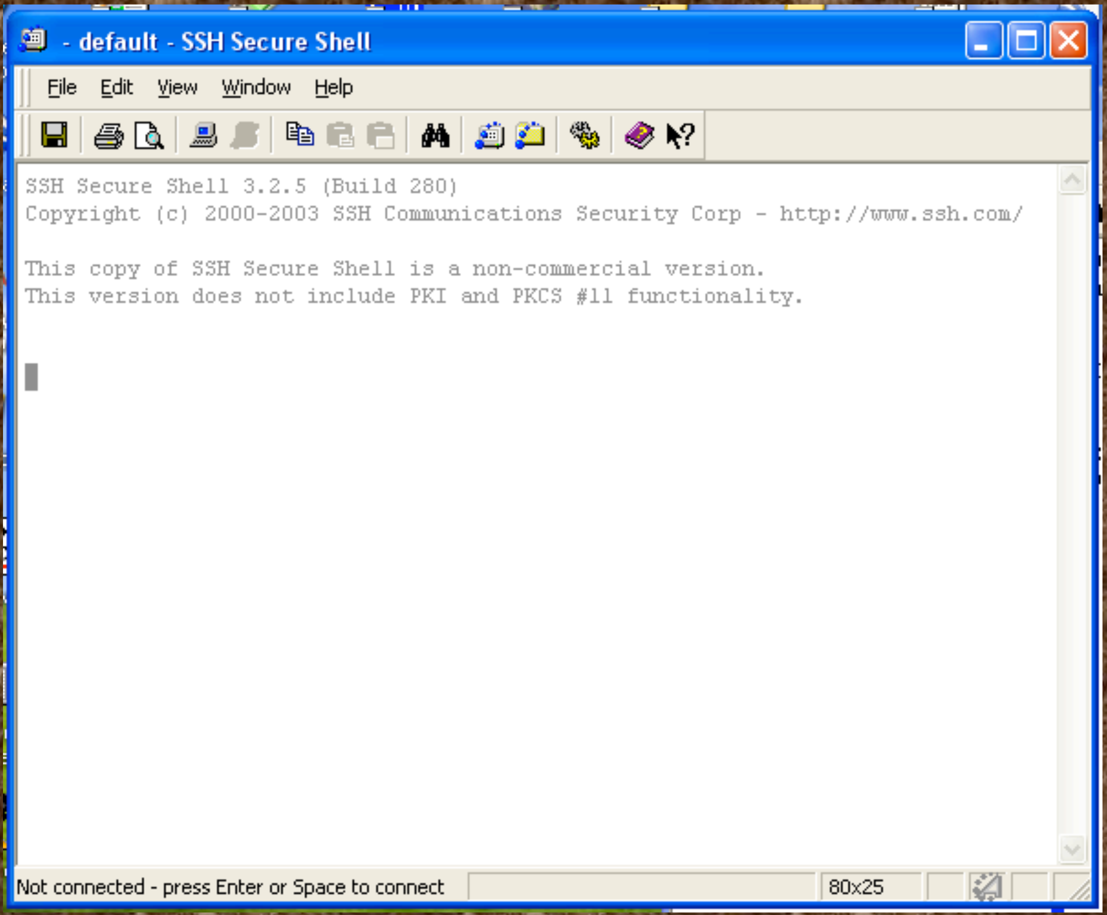
On the PC it is a few more clicks, but first we need to install two programs SSH Secure Shell Client and Exceed (part of the Hummingbird package).



Double click on exceed (it will start up and put an icon in the tray, it does not have a window).

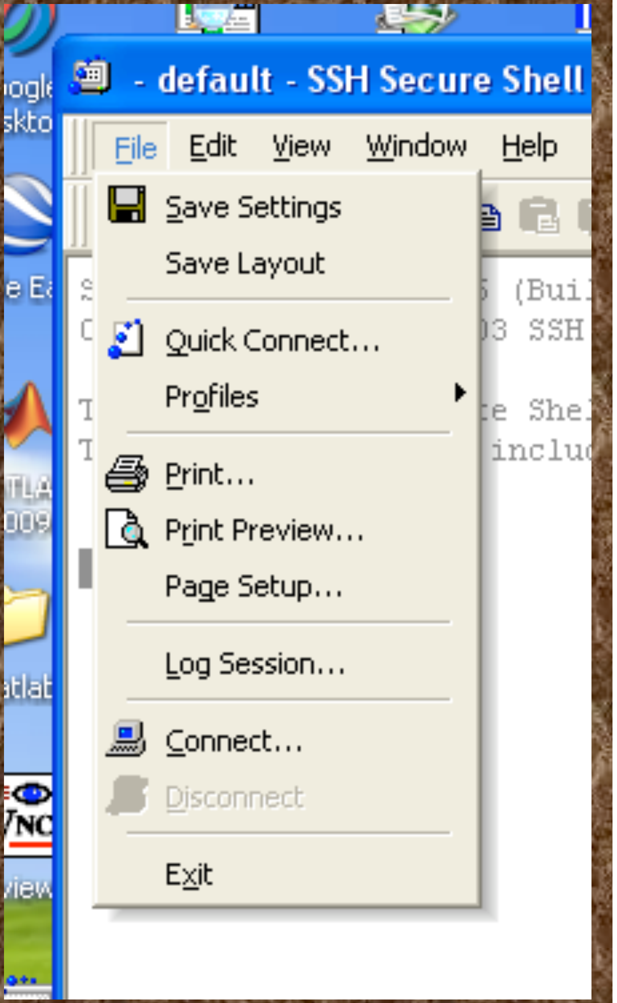
Double click on SSH Secure Shell Client

You will get this window (left). Now we have to connect to a machine. Click on File and then connect.

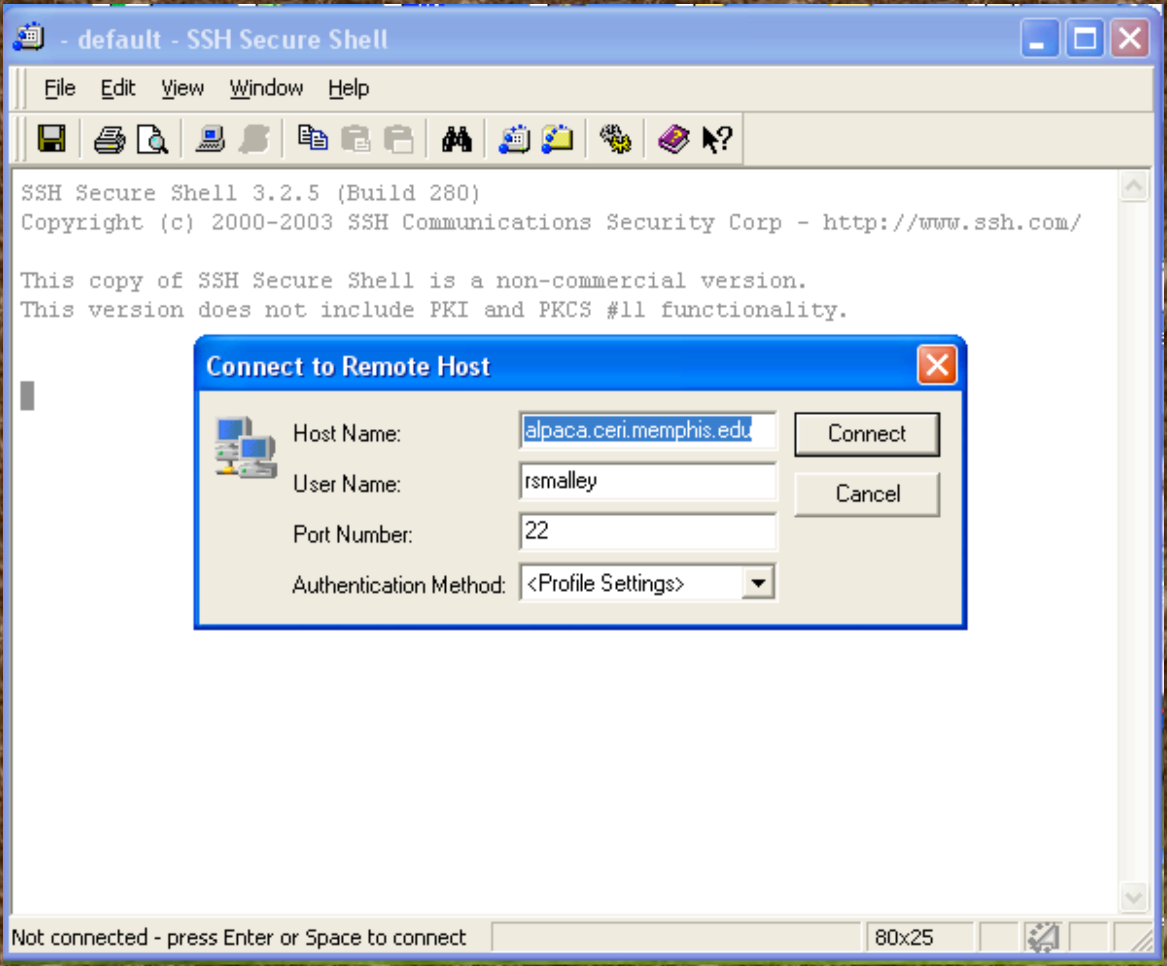


```
SSH Secure Shell 3.2.5 (Build 280)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.
```

Not connected - press Enter or Space to connect 80x25



This brings up the connect dialog. Put in the host name you want to connect to and your username. Leave the other stuff alone (default). Click connect.



```
SSH Secure Shell 3.2.5 (Build 280)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.
```

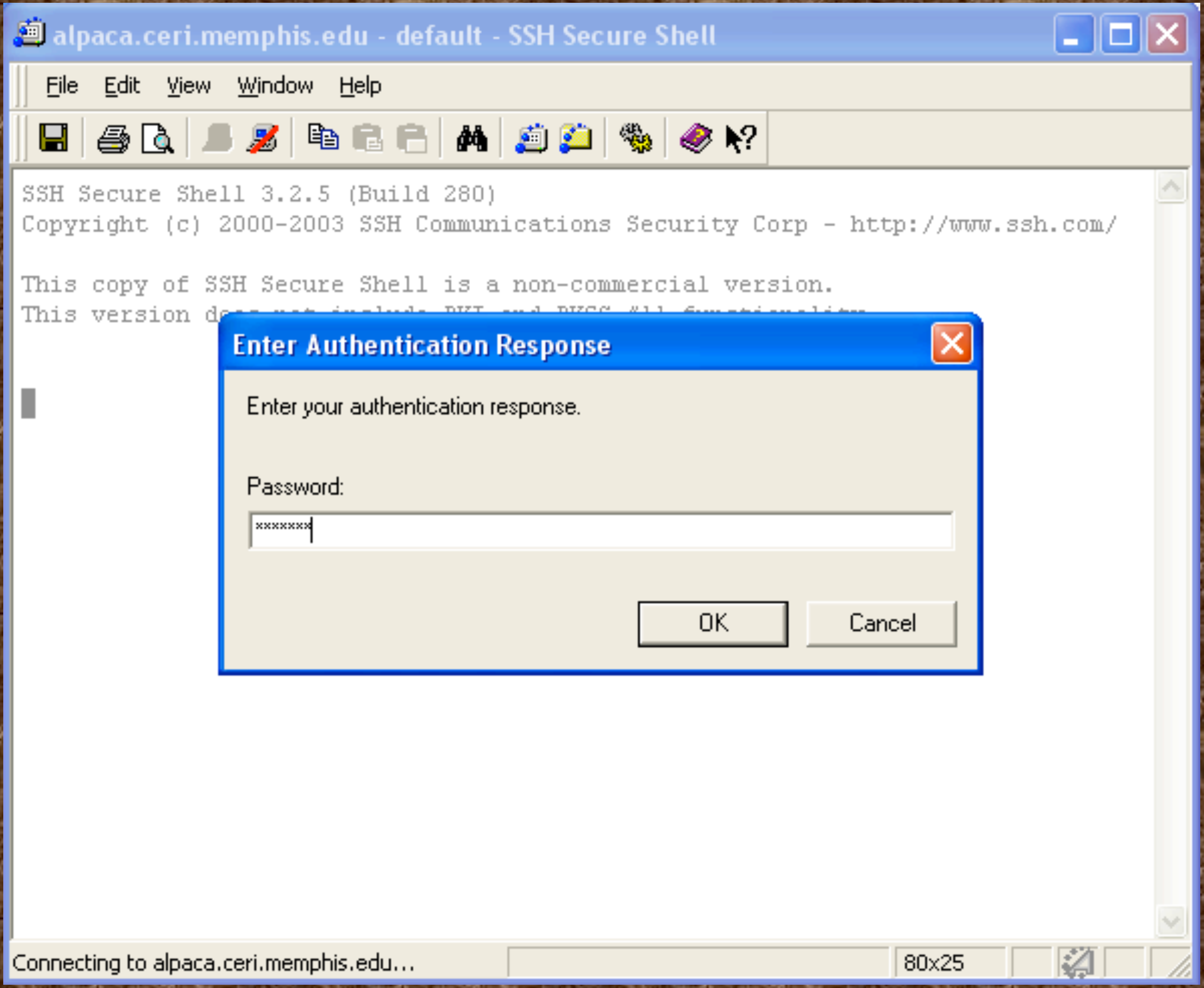
Connect to Remote Host

Host Name:	alpaca.ceri.memphis.edu
User Name:	rsmalley
Port Number:	22
Authentication Method:	<Profile Settings>

Not connected - press Enter or Space to connect

80x25

It will now ask for your password.



```
SSH Secure Shell 3.2.5 (Build 280)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version d
```

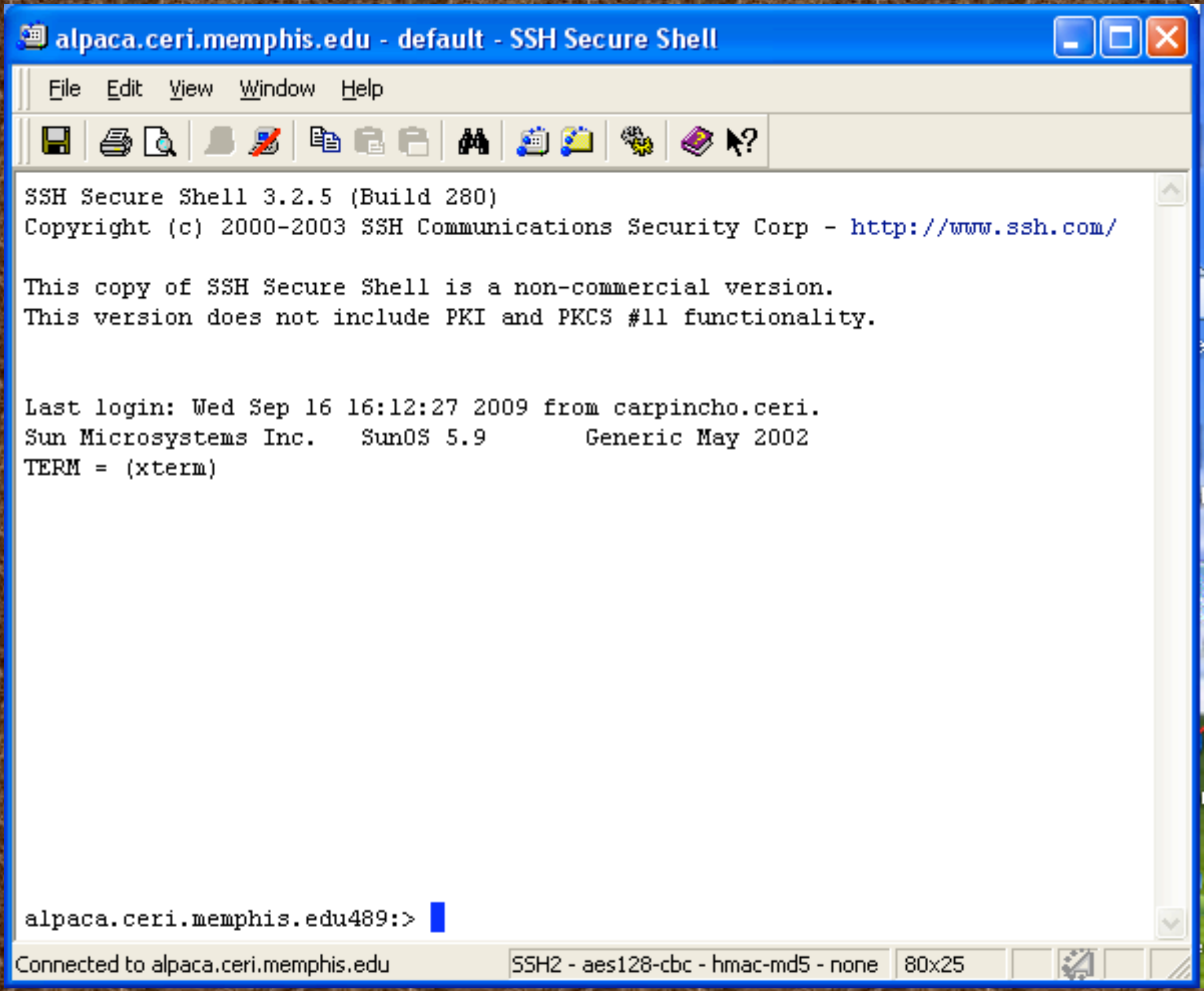
Enter Authentication Response

Enter your authentication response.

Password: *******

Connecting to alpaca.ceri.memphis.edu...

And we are finally connected.

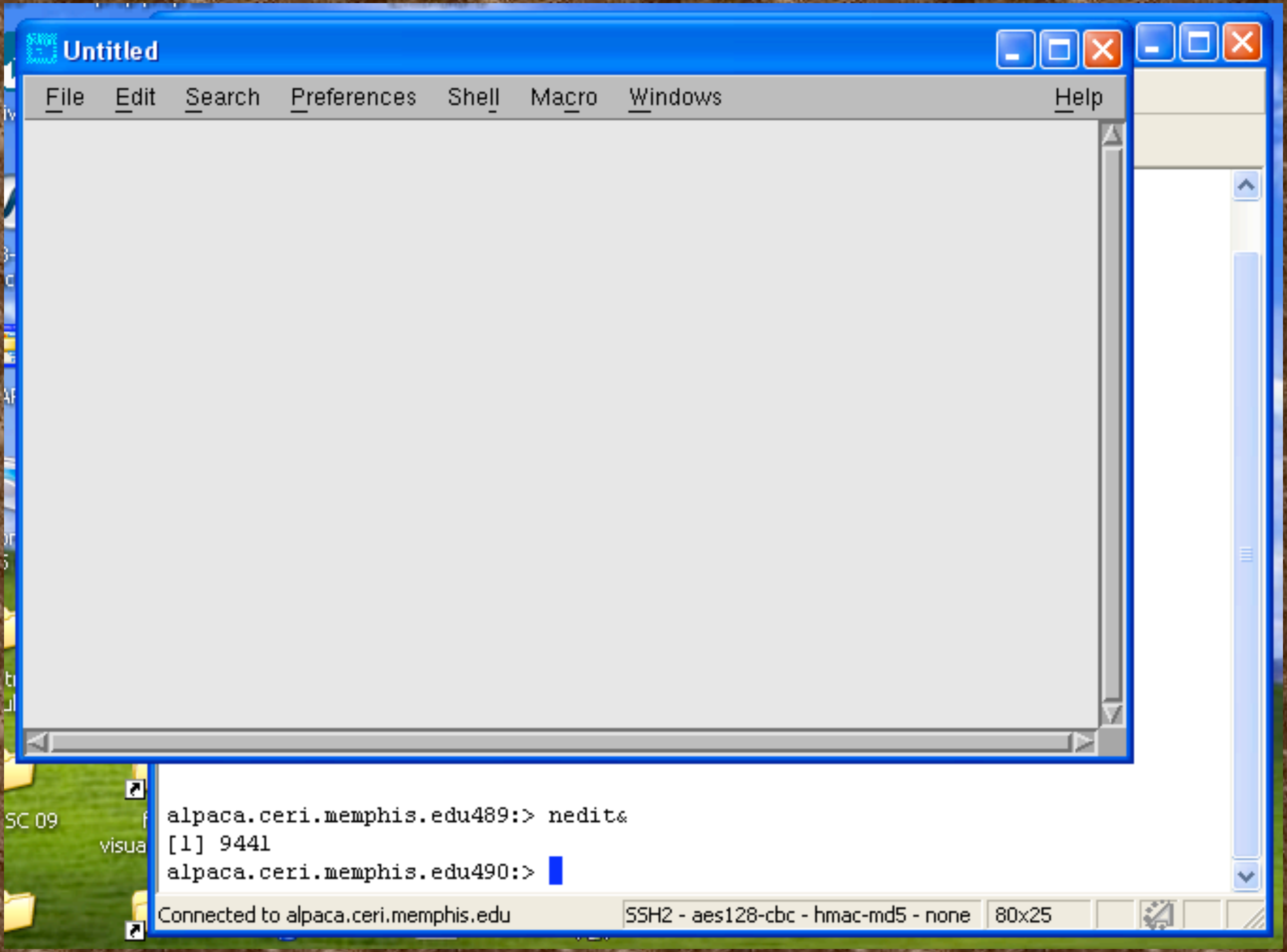


alpaca.ceri.memphis.edu - default - SSH Secure Shell
File Edit View Window Help

```
SSH Secure Shell 3.2.5 (Build 280)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.


Last login: Wed Sep 16 16:12:27 2009 from carpincho.ceri.
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
TERM = (xterm)










alpaca.ceri.memphis.edu489:>
```

```
Connected to alpaca.ceri.memphis.edu    SSH2 - aes128-cbc - hmac-md5 - none    80x25
```

Start nedit in the background.



```
alpaca.ceri.memphis.edu489:> nedit&
[1] 9441
alpaca.ceri.memphis.edu490:>
```

Connected to alpaca.ceri.memphis.edu   SSH2 - aes128-cbc - hmac-md5 - none   80x25

Text Editing

BASICS OF THE UNIX/LINUX ENVIRONMENT

Text Editing Options

Mouse-driven options

nedit: this X-window GUI text editor allows interactive mouse or keyboard driven text manipulation; colored text and auto-recognition of various standard scripting and programming languages is helpful for debugging scripts and code; appears to be a student favorite at CERI and is available on the Unix system.

Text Editing Options

Mouse-driven options

<u>emacs</u>: a less sleek looking GUI text editor (available at CERI) that allows interactive mouse or keyboard driven text manipulation; it is very powerful and is an old favorite of computer programmers.

Text Editing Options

Keyboard-driven options

vi or vim: this non-GUI text editor relies primarily on keyboard driven text manipulation; steep learning curve but very powerful; vim - colored text and auto-recognition of various standard scripting and programming languages is helpful for debugging scripts and code.

Found on ALL Unix systems.

Text Editing Options

Keyboard-driven options

pico: a pared down non-GUI text editor very similar to the email program pine. If you don't know what pine is, use nedit instead.

nedit or vi/vim.

nedit is available on the CERI unix machines because Bob and Mitch have installed it.

nedit has a shallow learning curve (execute it and start using!).

nedit or vi/vim.

vi and vim are available standard on all Unix and Unix-like systems.

vi and vim are harder to learn.

vi and vim are much more powerful than nedit.

*note to OSX users, nedit can be downloaded and installed on OSX but you need to be sys admin and know what you are doing….it is not a simple dmg unpack. Xcode is a similar but more powerful editor for code development.

to start nedit

```
%nedit &
```

Which introduces another Unix feature - the "&".

When the "&" is placed at the end of a command line it opens the program in the background so that you can continue to use the terminal window.

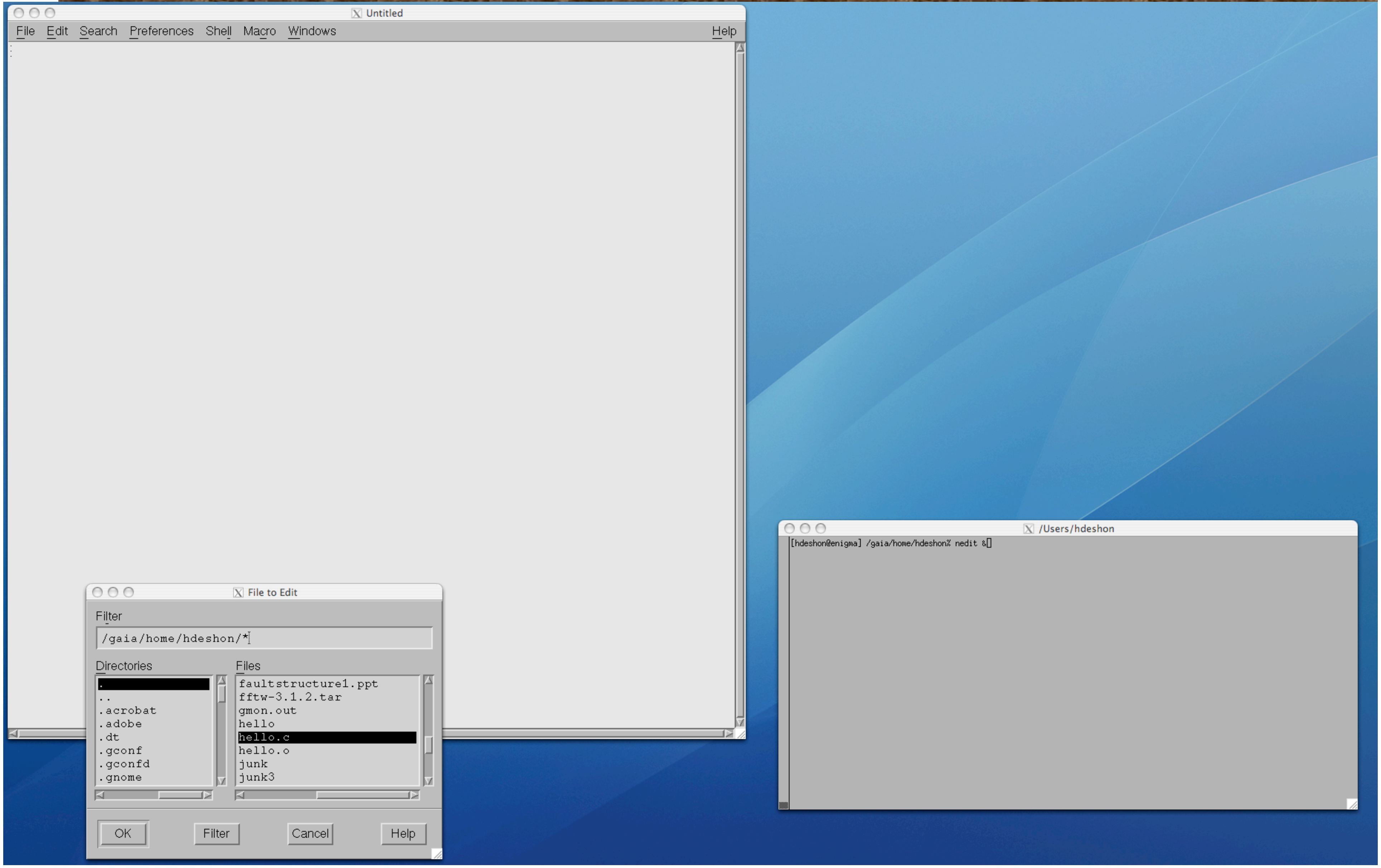
&

This is a general feature.

So if you have a program that will take 10 minutes to run and is putting its output into a file (not the screen), you can run it with the & at the end and it will go off and do its thing in the "background" and you can continue working in the window.

This was a much more important before the days of window based GUIs.

This is what it looks like
(using a mac that is ssh'd into the suns)



```
/gaia/home/hdeshon/*
```

Directories:

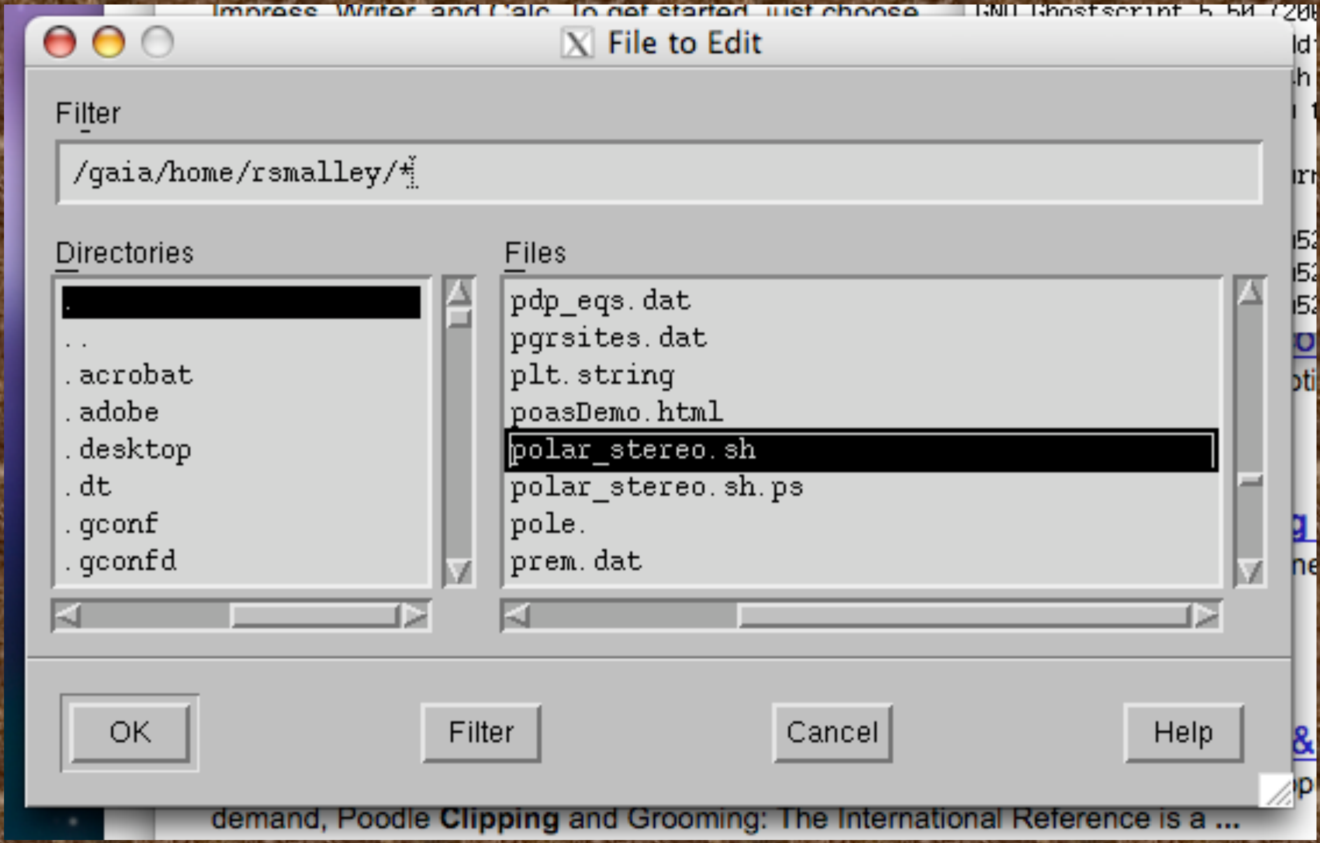
```
.
..
.acrobat
.adobe
.dt
.gconf
.gconfd
.gnome
```

Files:

```
faultstructure1.ppt
fftw-3.1.2.tar
gmon.out
hello
hello.c
hello.o
junk
junk3
```

```
[hdeshon@enigma] /gaia/home/hdeshon% nedit &
```

Works similar to WORD. File/open – get dialog box. Select file to open.



File to Edit

Filter

```
/gaia/home/rsmalley/*
```

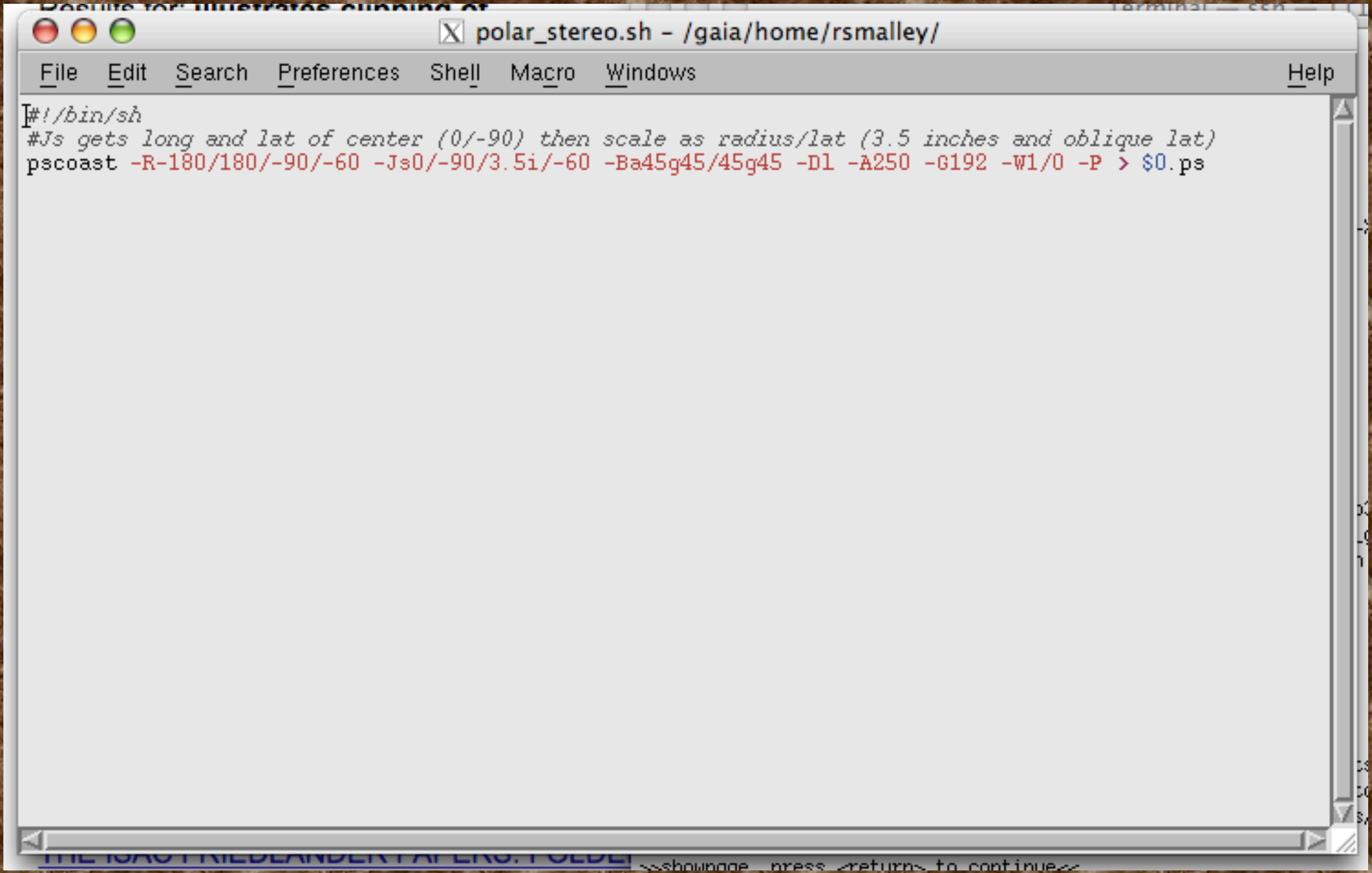
Directories

```
.
..
.acrobat
.adobe
.desktop
.dt
.gconf
.gconfd
```

Files

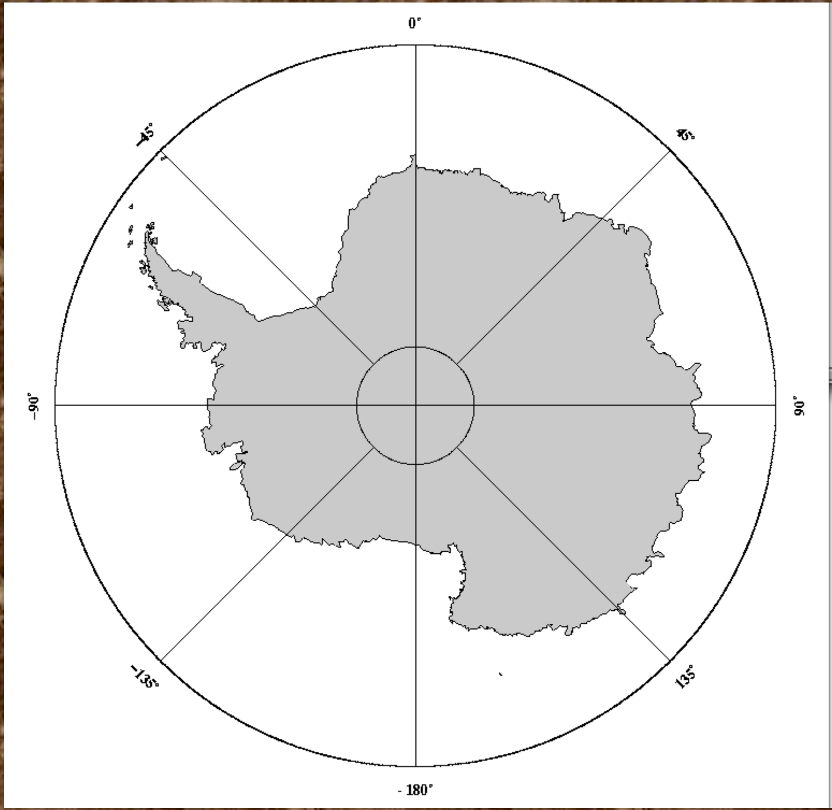
```
pdp_eqs.dat
pgrsites.dat
plt.string
poasDemo.html
polar_stereo.sh
polar_stereo.sh.ps
pole.
prem.dat
```

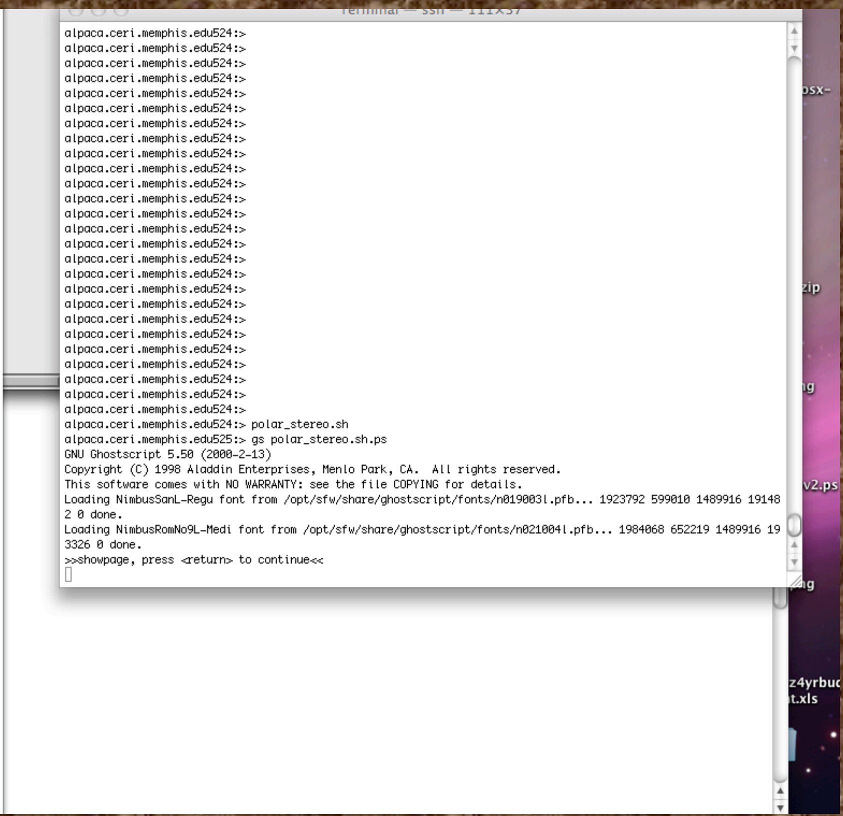
This is the file. It is a shell script (bourne shell - sh). It makes a map using the GMT package.



```
#!/bin/sh
#Js gets long and lat of center (0/-90) then scale as radius/lat (3.5 inches and oblique lat)
pscoast -R-180/180/-90/-60 -Js0/-90/3.5i/-60 -Ba45g45/45g45 -Dl -A250 -G192 -W1/0 -P > $0.ps
```

Here's what you get when you run it.





```
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:>
alpaca.ceri.memphis.edu524:> polar_stereo.sh
alpaca.ceri.memphis.edu525:> gs polar_stereo.sh.ps
GNU Ghostscript 5.50 (2000-2-13)
Copyright (C) 1998 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
Loading NimbusSanL-Regu font from /opt/sfw/share/ghostscript/fonts/n019003l.pfb... 1923792 599010 1489916 19148
2 0 done.
Loading NimbusRomNo9L-Medi font from /opt/sfw/share/ghostscript/fonts/n021004l.pfb... 1984068 652219 1489916 19
3326 0 done.
>>showpage, press <return> to continue<<
```