Cshrc and Login Files

Overview

The C Shell, `csh`, uses several configuration files. When it first starts up, it performs the commands in the file `~/.cshrc` (remember that `~` is your home directory). If it is a login shell[1] then it reads the systemwide configuration file `/etc/.login`, then `~/.login` and finally `~/.logout` when you logout.

Opinions vary about what should go in each of these files. Normally, you should use `.login` to setup your terminal and `.cshrc` for other settings. Since `.cshrc` gets run even for noninteractive shells, it shouldn't print anything to the screen, or make settings that will cause trouble for shells scripts. We'll see that one way to handle this is to test if the shell is being used interactively, and do different things accordingly.

In the math department, the systemwide file checks your disk quota, establishes default values for your terminal type, prints the message of the day, and tells you if you have mail.

First, we'll look at some simple example files. Then, we'll talk about other settings/options, and give some examples of other things you can have in these files. Finally, we'll look at the default setup for the Berkeley math department, and an example of another set of files for use here.

- Before you start, read the `csh` man page, to learn about the shell's syntax, built-in commands, and predefined shell variables and what they configure.
- Note that the exact contents of files will depend not only on your preferences, but on the particular version of Unix and system configuration.
- I did reformat some things a bit to make them fit, splitting long lines with backslashes. Everything should still work, but if you have trouble, convert them back.
- Both `^[` and `^G` are control characters. Generally, you can insert them into a file by typing control-V and then the control character you want.
- Some people start a windowing system from their `~/.login`, but I think this leads to more trouble than it's worth. As our Suns become converted to Solaris with graphical logins, it's pointless anyway.
- See `http://www.perl.com/perl/versus/csh.whynot` to understand why you shouldn't attempt complicated shell scripts in `csh`.

— *Thomas Insel, April 1999*

[1]Login shells show up in your `ps` listing with names that begin with a `-`, and are typically the ones that you type into, not ones that run shell scripts, etc.

Not all interactive shells are login shells, and it's complicated to explain how you get each type. If you telnet or rlogin interactively, you should get a login shell. If you run `xterm -ls` you will get a login shell, but normally when you open a shell window or give an rsh command, it isn't a login shell.

Simple .cshrc

First, we make settings for all shells. Note that csh ensures that the shell variable path is kept in synch with the environment variable PATH. It does the same for home/HOME, term/TERM, and user/USER. We set environment variables: EDITOR and PAGER are used by many programs, EXINIT configures ex and vi, MORE configures more. See the appropriate man pages for more information. The umask means that group and other get no permissions by default. Setting noclobber ensures that redirection won't erase existing files.

Then, we exit if not interactive, so that we don't bother with aliases and such for shell scripts and rsh sessions. Also, .cshrc mustn't write to the screen in these circumstances.

Finally come settings for interactive shells. We tell the shell to check for new mail every minute. Note how we seperate the first word of the hostname when setting the prompt.

```
umask 077
limit coredumpsize 0
set path = (/bin /usr/bin /usr/local/bin $HOME/bin)
set noclobber
setenv EDITOR    vi
setenv PAGER     less
setenv EXINIT    'set ai showmatch'
setenv MORE      -d

if (! $?prompt) exit 0 # --- quit if not interactive.

mesg y
set mail = (60 /var/mail/$USER)
unset ignoreeof

set hostname = `hostname | sed -e 's/\..*//'`
set prompt = "${user}@${hostname} \!% "

alias rm 'rm -i'
alias cp 'cp -i'
alias mv 'mv -i'
```

Simple .login

This sets up terminal characteristics:

1. Use `set noglob` to turn off wildcard matching, so that `?` doesn't cause trouble.
2. Run `tset`. It first tries to determine the terminal type. The base value is a function of how you connect (see `/etc/ttytab`). Then, it applys substitutions specified with the `-m` arguments, prompting the user if appropriate. Finally, tset initializes the terminal, and sets the `TERM` and `TERMCAP` environment variables. You should read the man page.
3. Turn wildcards back on.
4. Use `stty` to set terminal options. Here, we tell it to use the default settings for a CRT (as opposed to a Teletype, I guess) and to expand tabs to spaces.
5. Optionally, the end of this file would be a good place to put informative commands that you'd like to run when you log in, perhaps `uptime` or `users` to see how busy the system is.

```
set noglob
eval ‘tset -s -mannex:vt100 -mcon80x25:vt100 -mnetwork:?$term‘
unset noglob
stty crt -tabs
```

Shell Variables

The shell has many predefined shell variables. Some contain useful information. Others control the shells behavior. We'll talk about these now. Use the `set` to see the current settings. See the `csh` man page for more information. Lists should begin and end with parentheses and are seperated by spaces.

cdpath — a list of directories to be searched by `cd`, `chdir`, `pushd` commands if their argument doesn't match a subdirectory of the current directory.

echo — if set, echo commands just before executing them. Use this for debugging, or to see how wildcards expand.

filec — if set, enable filename completion with control-D (lists available completions) and ESC (completes as much as possible uniquely).

  fignore – a list of filenames to ignore when attempting filename completion.

  nobeep – if set, don't beep for ambiguous filename completion attempts.

histchars – set to a string of two characters. The first replaces `!` for history commands and the second replaces `^` for quick substitutions.

history — how many lines of past commands to remember.

  savehist — how many lines of history to save to disk.

ignoreeof — if set, you can't use control-D to exit or logout.

mail — a list of files to check for mail. If the first word is a number, how often to check (in seconds).

noclobber — if set, redirection with `>` won't erase an existing file.

noglob — if set, don't expand wildcards.

notify — if set, the shell will notify immediately when a job finishes. Otherwise, only when printing a prompt.

path — list of directories to search for commands.

prompt — essentially, `csh` sets this to `%` for interactive shells and leaves it unset (i.e., `$?prompt == 0`) if the shell is noninteractive. You can set it to your favorite string, noting that a `!` will be replaced by the current command number.

time — controls if and how the shell reports time and memory useage when running programs. See the man page.

verbose — echo commands after history substitution.

Stuff

Other things you might want to set:

- Use the mesg command to control whether other users can write to your terminal via write, send talk requests, etc. Use mesg y to enable this and mesg n to disable it.

- Use biff y in your .login if you want to be notified of new mail immediately, not just when waiting at a shell prompt.

- Solaris and Linux supports non-English languages, which can be controlled via the LANG environment variable. It's not supported by everything, and you'll need to work out some character set issues, but try:

  ```
  setenv LANG es; date
  ```

- If you set TZ to a time zone abbreviation (e.g. CDT or GMT+3), the system will report the time in that area, instead of local time.

- TERM, TERMCAP, TERMPATH, ...

- Use the limit builtin command to limit resource usage (to guard against runaway programs, etc.). The defaults are something like:

  ```
  limit cputime      unlimited
  limit filesize     unlimited
  limit datasize     2097148 kbytes
  limit stacksize    8192 kbytes
  limit coredumpsize 0 kbytes
  limit descriptors  64
  limit memorysize   unlimited
  ```

- Call umask with a three-digit octal number to set the user file-creation mode mask. There is one digit each for the user, group, and others. Each digit is the sum of the read (4), write (2) and execute (1) permissions that should not be allowed in newly created files.

- The environment variable PRINTER is used by lpr and lpq and can be set to something like hp1.

- Use stty to make terminal settings (can change the size manually if necessary, set backspace/erase, and so on).

- Aliases are mostly for convenience, and because shell scripts can't modify variables in the parent shell's environment. Define aliases like:

  ```
  alias x '\!*'
  ```

where the `\!*` gets replaced by the commands arguments. An alias can call other aliases, but not itself.

To see all current aliases type `alias`. There's an `unalias` command, too.

Tricks

If you want to have the current directory in your prompt try (or see `tcsh`):

```
alias np set prompt='${user}@${hostname}:${cwd}%\ '
np
alias cd chdir \!:\* \; np
alias pd pushd \!:\* \; np
alias pp popd \!:\* \; np
```

Here's how you can check for a particular machine or operating system (note also how we add a directory to the end of the existing path):

```
set arch = `arch`

if ($arch == sun4 && `uname -r` =~ 4.* ) then
    # SunOS on Sparc
    set path = ( $path $HOME/sunos-bin )
else if ($arch == sun4 && `uname -r` =~ 5.* ) then
    # Solaris on Sparc
    set path = ( $path $HOME/solaris-bin )
else if ($arch =~ i*86 `uname` == Linux) then
    # Linux on Intel
    set path = ( $path $HOME/linux-bin )
endif

unset arch
```

Adapted from `std.login`, to tell people that finger you where you last logged in:

```
echo Last logged in to `hostname` at `date "+%r %a %D"` > .plan
```

Some things don't need to be set each shell. For example, environment variables are inherited so they need only be set once. However, they can't just be put in `.login`, since they should be set for nonlogin or even noninteractive shells. A technique to deal with this, adapted from the math department setup, is to use something like:

```
if ( ! $?ONCEONLY ) then
    setenv ONCEONLY 1
    setenv MANPATH  /usr/local/man:/usr/man:/usr/share/man
endif
```

Berkeley Files

Here, we describe the behavior of the files attached in the Appendix, which are attached at the end, and are current as of April 5, 1999.

Note that `std.cshrc`, `std.login`, and `std.logout` are meant to be included from your `.cshrc`, `.login`, and `.logout` files, not stand on their own. The idea is that the system staff can change these files as software is installed or the system is reconfigured, and everything *should* work transparently to us.

.cshrc

Essentially, this file sources `std.cshrc`, which does:

- Initializes environment variables including `HELPPOOL`, `MANPATH`, `NNTPSERVER`, `ORGANIZATION`, `MORE`, `LD_OPTIONS`, `TERMINFO`, `MacalayPath`, and `XKEYSYMDB`. (**THESE NEED EXPLAINING.**)
- Sets basic `csh` settings: `noclobber`, command history, filename completeing, mail checking, `limit` and `umask`.
- Sets the path based on machine and operating system.
- Sets your prompt.
- Sets aliases: `ts`, `matlab`, `math`, and `x`.

The `.cshrc` also creates aliases for `logout`, and aliases `rm`, `cp`, and `mv` to not delete files without asking you first. It sets up a `back` command that you can use to undo the most recent `cd`. Finally, it changes the titlebar of your window if you're using SunView (you aren't).

Note that `std.cshrc` exits halfway through if the shell is not interactive, anything that comes after the "`source /usr/local/lib/std.cshrc`" line of your `.cshrc` will not be executed for noninteractive shells.

.login

All but one line is commented out, and that line sources `std.login`, which does:

- Works around some bugs in the terminal database and then runs tset to setup the terminal.
- Sets the titlebar of your window if appropriate.
- Executes a `.reminder` file if it exists (this is used to make sure you set your username at first login, for example).
- Tries to set your `DISPLAY` (but messes up if you've connected from a Sun 3/50 running as an X Terminal).

Of the other options offered in `.login` that we haven't already discussed, a few don't seem to be used on our computers (so don't bother setting them). You don't need to set `XENVIRONMENT`, since `std.xinitrc` will read `~/.Xdefaults` automatically if it exists, and you don't need to set anything to do with NeWS or SunView.

.logout

Sources `std.logout`, which deletes some unneeded files in your home directory: TEX logs, DVI files, and backup files ending in `~`. This was useful a few years ago when our disk quotas were orders of magnitude smaller.

Linux

As of this writing, the Linux computers (e.g. `koebe.math`) are configured differently – `std.cshrc` is a link to `/etc/csh.cshrc` which `tcsh` sources automatically, and there is no `std.login`. To work around this, you can add a simple test to your `.login` and `.cshrc` files:

```
if (`uname -s` == Linux) then
    # do whatever needs doing.
else
    source /usr/local/lib/std.login
endif
```

I expect that this will change as Intel machines running Linux become more supported on the department network.

Example Berkeley Files

~tinsel/.cshrc

```
source /usr/local/lib/std.cshrc

setenv PAGER less
setenv EXINIT 'set ai nomagic tabstop=8 wrapmargin=10 showmatch'
unset ignoreeof

if ($?tcsh) then
    set rmstar correct=cmd
    set prompt="%n@%m:%c4> " prompt3="$prompt%R (y|n|e)?"
else
    set prompt="$user@$hostname% "
endif

alias jaka 'ssh2 jaka.ece.uiuc.edu'
```

~tinsel/.login

```
if ($TERM == annex) setenv TERM vt100
if ($TERM == con80x25) setenv TERM vt100
if ($TERM == vs100) setenv TERM xterm
if ($TERM == network) setenv TERM vt100
if ($TERM == linux && `uname` != Linux) setenv TERM vt100

source  /usr/local/lib/std.login
```

~tinsel/.logout

I don't have one, because I'm tired of all my DVI files disappearing when I close a window.

tcsh

The tcsh shell is essentially compatible with csh but adds many extra features. The manual page is essential reading, but I'll summarize some of what you can do.

- Command line editing when edit is set. Use left/right arrows to edit, and up/down arrows to scroll through your history. Full vi, emacs, and customizable key bindings are available through the bindkey command.
- Command line completion with control-D and TAB. Highly customizable through many shell variables, and the complete command (e.g., you can have ftp and telnet telnet complete from a list of computers you usually connect to, etc.).
- Spelling correction for commands and/or filenames. Set correct to cmd or to complete.
- Special aliases: beepcmd, cwdcmd, periodic, precmd that can run when the shell wants to beep the bell, when the current directory changes, every tperiod minutes, or before each prompt. Also, can have autologout for idle shells, and look for other users automatically with the watch and who variables. The sched builtin command allows you to schedule programs to run in the future (but only while you're logged in).
- More options for customizing your prompt. I like to see my username, computer, and directory, so I use:

  ```
  set prompt="%n@%m:%c4> "
  ```

- Supports localization, allows 8-bit filenames and Kanji if your OS supports them.
- Improved terminal management.

If you may use both csh and tcsh and want to use one .cshrc to configure both (instead of a .tcshrc and a .cshrc), you can test which shell is running with

```
if ($?tcsh) then
    echo "I am tcsh."
else
    echo "I am csh."
endif
```

Since xterm supports a special character sequence to change the titlebar, I use the following code with tcsh to remind me which window is which:

```
if ($TERM == xterm || $TERM == sun && "`tty`" != /dev/console) then
    alias precmd "echo -n '^[]0;${user}@${hostname}^G'"
endif
```

Appendix: Berkeley File Listings

.cshrc

```
############################################################################
#                                                                          #
# This is the standard .cshrc file.  See also /.cshrc and "help dotlogin" #
# It is read in once after the .cshrc file when you log in, and it is     #
# also read  every time you open a Sheltool or CMDTool under Suntools.    #
#                                                                          #
############################################################################

# Source the standard .login file - Do not edit the next line, unless
# you really know what are you doing.
source /usr/local/lib/std.cshrc

# This sets your terminals if you have a fixed one. Take the # outs and
# set them the way you need:
#set dialterm = vt100
#set plugterm = wy60
#set fastswitch = wy60
#set switch1200 = vt100
#set switch2400 = vt100

#set cdpath = (/your favorites dir here,second favorite,etc)

# This sets up lots of useful aliases.
#
# Executes some dangerous commands in interactive mode.
alias cp cp -i
alias mv mv -i
alias rm rm -i

# Log out any way you want:
alias bye logout
alias adios logout
alias logoff logout
alias quit logout

#Directory forward and back
alias cd 'set olddir=`pwd`; chdir \!*;'
alias back 'set back=${olddir}; cd ${back}; unset back;'

#Window headers
```

```
if ($?TERM) then
    if ($TERM == sun && "`tty`" != /dev/console ) then
        alias header echo -n "^[]l $hostname "'${cwd}''^[\\'
        alias lheader echo -n "^[]L $hostname "'^[\\'
        lheader
        header
alias cd 'set olddir=`pwd`; chdir \!*; header'
    endif
endif
```

.login

```
############################################################################
#                                                                          #
# This is the standard .login file.  See also /.login and "help dotlogin"  #
# It is read in once after the .cshrc file when you log in. It is NOT read #
# when you open a Sheltool or CMDTool under Suntools.                      #
#                                                                          #
############################################################################

# source the standard .login file - Do not edit the next line, unless
# you really know what are you doing.
source /usr/local/lib/std.login

# The next few lines set a few personall stuff for your account. Edit it
# to your taste, and remove the # character from the line. They set the
# prompt, editor, environment for X11, your name, ...

# Set your prompt here and in .cshrc if you want it in a different way:
#set     prompt = "$hostname >"
# Uncoment this line out if you want your editor to be Emacs (default is VI)
#setenv EDITOR          emacs
# Uncomment the next line only if you have your own X setup.
#setenv XENVIRONMENT    ~/.Xdefaults
# Write your name in the next line and uncomment it out.
#setenv NAME            "John Q. Public"
# Uncomment the following line if you want a specific printer
#setenv PRINTER         lw0
#setenv LESS            QpmPmFoP
#setenv RNINIT          -e -h +hfrom +horganization +hdate -hdate-received \
                        +hsummary +hsubject
#setenv NEWSSERVER      used by Mathematica under NeWS.
#setenv WINDOW_PARENT   Used by Mathematica in Sun View.
```

```
#setenv SUNPSFONT       Used by Mathematica to find Screen font under SunView.
#setenv XPSFONT         Used by Mathematica to find Screen font under X windows.
# That will have to wait until we compile the Emacs with the right support
# for Suntools.
# This next one is important for people using DSTOOL:
#setenv LD_LIBRARY_PATH $OPENWINHOME/lib
# Use the variable below to set a specific font-path for "xdvi".
#setenv XDVIFONTS       /vol/moby/moby_a/tex82/common/fonts/%p_lw
# Uncomment this line if you want to use SUN terminfo (default is customized)
#setenv TERMINFO        /usr/share/lib/terminfo
# This next line extends the search path for TEXINPUTS files. The position
# of the $TEXINPUTS is absolutely essential, do not change it.
#setenv TEXINPUTS $TEXINPUTS":"$HOME/text/tex
# Set this one to your favorite .defaults file under SunView:
#setenv  DEFAULTS_FILE /usr/local/lib/menus/defaults
```

.logout

```
# source the standard .logout file
source /usr/local/lib/std.logout
```

std.cshrc

```
############################################################################
#                                                                          #
# This is the standard .cshrc file.  See also /.cshrc and "help dotlogin" #
# It is read in once BEFORE the .login file when you log in, and it is    #
# also read  every time you open a Sheltool or CMDTool under Suntools.    #
#                                                                          #
############################################################################

if ( ! $?ONCEONLY ) then
        #
        #  This section of the .cshrc file does things we only want done
        #  once, but want done even if there was no login (e.g. via rsh).
        #
        setenv ONCEONLY 1

        setenv  EXINIT     'set showmatch'
        setenv  HELPPOOL   /usr/local/help/ccs/cat:/usr/local/help/math/cat
        setenv  MANPATH    /usr/local/links/man:/usr/local/misc/man:\
             /usr/local/x11/man:/usr/man:/usr/lang/man:/usr/openwin/man:\
             /usr/share/man/ccs
        setenv  NNTPSERVER      agate
        setenv  VAXIMA          /usr/lib/mac
        setenv  ORGANIZATION    "U.C. Berkeley Math. Department."
        setenv  PAGER           more
        setenv  MORE            -c
        setenv  LD_OPTIONS      -L/usr/lang/SC0.0
        setenv  TERMINFO        /usr/local/misc/etc/terminfo
        setenv  MacaulayPath .:/usr/local/Macaulay/scripts/scriptsde:\
            /usr/local/Macaulay/scripts/scriptsmj:\
            /usr/local/Macaulay/scripts/scriptsms:/usr/local/Macaulay/source/
        setenv  EDITOR          vi
        #setenv EDITOR          emacs
        #setenv TEXEDIT         'emacs-19  +%d %s'
        setenv  EMACSTOOL       /usr/unsupported/bin/emacs
        setenv  OPENWINHOME     /usr/openwin
        setenv  DSTOOL          /usr/local/dstool
        setenv  DSTOOL_COLOR_DIR /usr/local/dstool/colormaps
        setenv  DSTOOL_DATA_DIR  /usr/local/dstool/data

        # Needed for various X applications (e.g. xemacs):
        setenv  XKEYSYMDB       /usr/local/x11/lib/X11/XKeysymDB
endif


# Set things which are useful to everybody:
#
set     noclobber   # To avoid accidentally overwriting a file by redirection.
set     ignoreeof   # The end of file (^D) doesn't cause logout.
set     history=100
set     savehist=$history
set     filec
set     fignore = (.o .out .arc .bak .dvi .aux .bbl .blg .lof .log \
                   .lot .idx .ilg .ind .toc .ps)
set     mail =  ( 2 /usr/spool/mail/$USER )

limit coredumpsize 0
limit core 0

umask   077

# The beginning of all PATH individially for each machine:
# /bin got to be in here because the NeXT's keep several important commands
# like sed, hostname, next, ... are all in /bin on the NeXT.
```

```
set     common_path =           ( ~/bin \
                                /usr/{ucb,bin} /bin \
                                /usr/local/ssh/bin \
                                /usr/local/ssh2/bin \
                                /usr/local/opie/bin \
                                /usr/local/misc/bin \
                                /usr/local/links/bin \
                                /usr/local/bin \
                                /usr/unsupported/bin )
set     arch = `arch`

# Sun 4's running SUN OS:

if ($arch =~ sun4 && `uname -r` =~ 4.* ) then
                set path =      ( $common_path \
                                /usr/{lang,hosts,games} \
                                /usr/X11/bin \
                                /moby/{gnu,micro,misc,mtools,x11r6}/bin \
                                /moby/sun_answerbook_1.5/bin \
                                . )

# Sun Sparc running Solaris 2.5:

else if ($arch =~ sun* && `uname -r` =~ 5.* ) then
                set path =      ( /usr/local/SUNWspro/bin \
                                /usr/local/bin \
                                /usr/ccs/bin \
                                /usr/openwin/bin \
                                /usr/local/x11/bin \
                                $common_path \
                                /usr/{lang,hosts,games} \
                                /usr/local/mtools/bin \
                                . )

# Linux:

else if ($arch == i686) then
                set path =      ( $common_path \
                                /usr/X11/bin \
                                . )

else
                set path =      ( $common_path \
                                . )
endif


unset   common_path

#
# If invocation is non_interactive, skip the rest of the .cshrc
#
if ( $user == 0 || ! ${?term} )     exit

set hostname = `hostname | sed -e 's/\..*//'`
set prompt = "$hostname \!->"

# This sets up lots of useful aliases.
#
alias   ts      'set noglob; eval `tset -s \!*`; unset noglob'

# Sets the alias for Matlab depending on the machine you are on:
if ($arch =~ sun4 && `uname -r` =~ 4.* ) alias matlab echo "Matlab is \\
available only on Solaris machines, please use the ones in room 708."

# Sets the alias for Mathematica depending on the machine you are on:
if ($arch =~ sun* && `uname -r` =~ 4.* ) alias math echo "\\
MATHEMATICA version 3 is available on Solaris-2 machines only.\\
"

alias   mathematica     math
alias   mathem          math

alias   top             top -I
alias phone "finger \!:1@berkeley.edu | sed 's/Community Profile Database//'"

# These next aliases are useful only if you log on from the CONSOLE.
if ("`tty`" != "/dev/console")  exit
alias   x       x11
alias   X       x11
alias   startx  x11
alias   xstart  x11
alias   x11 \
    'xinit `if (! -e ~/.xinitrc) echo /usr/local/misc/lib/std.xinitrc` \
    $hostname;kbd_mode -a;clear'
alias   x11color \
    'xinit `if (! -e ~/.xinitrc) echo /usr/local/misc/lib/std.xinitrc` \
    $hostname -- /usr/X11/bin/Xsun;kbd_mode -a;clear'
if (`constype` == gx) alias x11 x11color

alias   o       openwin
alias   openwin /usr/openwin/bin/openwin
```

std.login

```
##########################################################################
#                                                                        #
# This is the standard .login file.  It is read by a "login" c-shell    #
# after the .cshrc file when you log in. It is NOT read when you open   #
# a shell "window" or "rsh" a command.  See also "help dotlogin".       #
#                                                                        #
# NOTE WELL: This file is for things peculiar to the act of logging in  #
# interactively.  If you want to do something only once, such as setting#
# an environment variable, put it in the ONCEONLY section of the .cshrc.#
#                                                                        #
##########################################################################

#  One of the primary functions of the .login is to "condition" your terminal
#  by doing a "tset".  Here we also try to work around a couple of Solaris-2
#  bugs: there is no dtterm termcap entry and the xterm init sequence
#  does a clear-screen.
#
if ( ! $?term ) set term=`tset -`              # Make sure "term" defined
set oterm="$term"
switch ("$term")

    case xterm:
        set t="/usr/local/misc/lib/termcap"
        if ( "`uname -sr`" =~ "SunOS 5"* && -f "$t" ) setenv TERMCAP "$t"
        unset t
        breaksw

    case dtterm:
        if ( "`tset -`" == "" ) set term=xterm
        breaksw
endsw

set noglob
eval `tset -s -mnetwork:?$term -munknown:?$term`
```

```
unset noglob

if ( "$oterm" == dtterm  &&  "$term" == xterm ) stty -tabs
unset oterm

# This give a finishing touch to the set up, if you are using X windows.
# And also tell X windows where are you in terms of display.
#
if     ($TERM == xterm || $TERM == xterms || $TERM == sun-cmd) then
   set noglob
   if ( $?TERMCAP ) then
           setenv TERMCAP "$TERMCAP""ti=\E7\E[?47h:te=\E[2J\E[?47l\E8:"
   endif
   unset noglob
   if (! $?DISPLAY) \
   setenv DISPLAY `who am i | sed -e 's/.*(//' -e 's/[:.)].*$//'`.berkeley.edu:0
endif

# Tells people that finger you were are you logged in, but it needs to be
# shaped up a bit to not erase the .plan file of anybody.
#set    hostname = `hostname | sed -e 's/\..*//'`
# echo Logged in on $hostname at `date "+%r %a %D"` | cat - .planfinale >.plan

# print local reminder file if it exists:
if (-e .reminder && ! -z .reminder)     source  .reminder
```

std.logout

```
# clear    # was clearing over-quota messages

( cd ; \find . '(' -name '*~' -o -name '.*~' -o -name '*.log' -o \
   -name '*.dvi' ')' -a -exec rm -f {} \; & ) >& /dev/null

# This next line is useful to tell people where did you last logged on, but it
# it needs to be shapped up to not delete people's .plan file.
# (echo Logged out from $hostname at `date "+%r %a %D"` \
#   | cat - .planfinale >~/.plan &)
```

See Also

- Davey, Paul and Thyssen, Anthony. *Csh Startup Summary.* http://www.cit.gu.edu.au/~anthony/info/shell/csh.startup.faq

- DuBois, Paul. *Using csh and tcsh*. Campbridge: O'Reilly and Associates, 1995. See `http://www.primate.wisc.edu/software/csh-tcsh-book/` for related resources.
- Joy, William. *An Introduction to the C shell*. Berkeley: Department of EECS.
- And the manual pages for `csh` and `tcsh`.