A

Obtaining & Installing tcsh

This appendix describes how to obtain, build, test, and install *tcsh*. As I write, *tcsh* is at version 6.06. If a more recent version has been released, just substitute the new version number wherever you see 6.06 in those commands in which it appears below.

The first thing you should do is check whether or not an up to date *tcsh* is already installed on your system. If it is, you don't need to do anything except make *tcsh* your login shell. (See "Selecting a Shell" in Chapter 1, *Introduction*.) Otherwise get the current version and install it first.

Find out whether *tcsh* is installed and what its pathname is by asking your system administrator or by running this command:

```
% which tcsh
```

If *tcsh* is present, determine its version number using the following command. Use single quotes as shown but substitute the actual pathname if it's different than */bin/tcsh*:

```
% /bin/tcsh -c 'echo $version'
tcsh 6.00.02 (Cornell) 08/05/91 options 8b,nls,dl,al,dir
```

If the output of this command indicates that your *tcsh* is an old version (as it does here), you should get the current one.

Obtaining the Source Distribution

The *tcsh* source distribution is available on the Internet via anonymous FTP. Connect to *tesla.ee.cornell.edu*, change into the */pub/tcsh* directory, and transfer the file *tcsh-6.06.tar.gz* in binary mode. After you obtain the distribution, uncompress it and extract the files:

```
% gunzip < tcsh-6.06.tar.gz | tar xf -
```

Or, on System V systems:

```
% gunzip < tcsh-6.06.tar.gz | tar xof -
```

If you don't have *gunzip*, specify the filename without the *.gz* suffix as *tcsh-6.06.tar* when you get the distribution. The FTP server will uncompress the file for you. Then run one of these commands:

```
% tar xf tcsh-6.06.tar         (For non-System V systems)
% tar xof tcsh-6.06.tar        (For System V systems)
```

The *tar* command should produce a directory *tcsh-6.06* in your current directory. Change into that directory with *cd tcsh-6.06* and you're ready to begin the build process.

If you want to use a World Wide Web browser to obtain the distribution, use the following URL:

```
ftp://tesla.ee.cornell.edu/pub/tcsh/tcsh-6.06.tar.gz
```

Or, to obtain the uncompressed version:

```
ftp://tesla.ee.cornell.edu/pub/tcsh/tcsh-6.06.tar
```

After you obtain the file, unpack it using the instructions above.

Build the Distribution—Quick Instructions

If you're impatient, you can try a quick build to get going sooner.

If *imake*, *xmkmf*, and the X11 configuration files are installed on your machine, you should be able to build *tcsh* like this:

`% xmkmf`	Generate *Makefile* from *Imakefile*
`% make depend`	Generate dependencies (optional)
`% make`	Build *tcsh*

If you're not using *imake*, create a *Makefile* from the standard template and use it to build *tcsh*:

`% cp Makefile.std Makefile`	Copy *Makefile* from standard template
`% cp config/file config.h`	Create *config.h* from appropriate file in *config* directory
`% make`	Build *tcsh*

Whichever method you use, if the *make* command succeeds, you should have an executable *tcsh*; proceed to the section "Testing and Installing tcsh." Otherwise use the detailed instructions in the next section to build *tcsh*.

Build the Distribution—Detailed Instructions

Use the instructions in this section if the quick build doesn't work or if you want to review and perhaps modify the configuration parameters. Read the entire procedure described below before you try any of it.

The files that contain information about building *tcsh* are:

- *README*—the general readme file
- *README.imake*—the *imake*-specific readme file
- *FAQ*—the frequently-asked-questions list
- *Ported*—describes build flags for systems to which *tcsh* has been ported

I recommend that you browse through these files before proceeding.

Overview of the Build Process

Here's a summary of the steps involved in building *tcsh*:

- Decide where you want to install *tcsh*.
- Configure *Makefile*. If you're using *imake*, the *Makefile* is generated from *Imakefile* and *imake.config*; otherwise it's created from *Makefile.std*.
- Configure *config.h*. This file contains system-dependent configuration flags used at compile time. It's created for you automatically if you're using *imake*, otherwise you create it from one of the files in the *config* directory.
- Configure *config_f.h*. This file contains options that turn on or off various *tcsh* features.
- Compile *tcsh*.

As you get set up to build *tcsh*, you may need to make changes to one or more of the files just mentioned. Use the following procedure to save a copy of any file you need to modify:

```
% cp file file.orig              Save copy of original file
% chmod 644 file                 Make working copy writable so you can modify it
```

That way you still have the original file to look at for reference as you modify your working copy.

Choose an Installation Directory

Before you build *tcsh*, think about where you're going to install it. The default installation directory is */usr/local/bin* but you can override it. For instance, I install *tcsh* in */bin* so I can use it even when the */usr* file system is unmounted.

tcsh is best installed in one of the directories in your system's standard search path, to make it easy for everyone on your system to use it. If you don't have permission to install files into any of those directories, ask your system administrator to install *tcsh* for you.

There are two aspects of deciding where to install *tcsh* if you don't want to use the default directory:

- The build procedure compiles a pathname into the *tcsh* binary so that *tcsh* knows how to set the value of the *shell* variable properly when it starts up. You need to determine what pathname to use.
- The install commands in the *Makefile* must know where to put the *tcsh* binary, so you need to tell those commands what directory to use.

In many cases, the directory used in the compiled-in pathname and for the install commands is the same. For example, you might compile a pathname of */bin/tcsh* into *tcsh* and then install the resulting binary into */bin*.

However, it's possible that you might want to configure the two directories to be different. You might want the stability of being being able to reference *tcsh* using a fixed name such as */bin/tcsh* but have the freedom to locate the actual binary wherever you want, such as */usr/local/new/tcsh*. This can be done by using */bin/tcsh* as the compiled-in pathname and installing *tcsh* in */usr/local/new*, then making */bin/tcsh* a symbolic link to */usr/local/new/tcsh*. Or you might want to draw a distinction between the compiled-in pathname and the location in which *tcsh* is actually installed if your systems run in an environment that uses NFS or AFS to share file systems over a network.

Configure the Makefile

The *Makefile* directs the build process by generating the commands needed to compile the intermediate object files and the final *tcsh* executable.

If you're using *imake*, the *Makefile* is generated from *Imakefile* and *imake.config* by running *xmkmf*. Take a look at *imake.config* to see if you want to make any changes. If you want to change the pathname that gets compiled into *tcsh*, define `TcshPath`. For example, to use */bin/tcsh*, add this line:

```
#define TcshPath /bin/tcsh
```

To change the directory used by the installation commands, define `DestBin`:

```
#define DestBin /bin
```

The manual page is installed by default as */usr/local/man/man1/tcsh.1*. If you want to change this, define `DestMan` as the installation directory and `ManSuffix` as the extension used for the file in that directory. For example, to install *tcsh.man* as */usr/man/mann/tcsh.n*, add these lines to *imake.config*:

```
#define DestMan /usr/man/mann
#define ManSuffix n
```

After you've looked through *imake.config* and made the appropriate changes, create the *Makefile* and generate the source file dependencies:[1]

`% xmkmf`	Generate *Makefile*
`% make depend`	Generate dependencies (optional)

If you're not using *imake*, the configuration process is different. First create a *Makefile* to work with by copying the template *Makefile.std*:

`% cp Makefile.std Makefile`	Copy working *Makefile* from *Makefile.std*
`% chmod 644 Makefile`	Make it writable

Then edit *Makefile* to choose the appropriate configuration parameters for your system. The *Makefile* itself has a lot of information about the settings for different systems, and you can also read *Ported* to see what special flags might be necessary for your machine. (The systems for a given vendor do not necessarily all appear together in *Ported*; be sure to look completely through it to find the best match for your system.)

The most important configuration parameters are listed below. Make sure you look at the possible settings in the *Makefile* and select those which are most appropriate for your system:

`CC` The C compiler

`DFLAGS` `-D`'s and `-U`'s to pass to the C compiler

`LDFLAGS` Loader (linker) flags

`LIBES` Link libraries

`CFLAGS` Special flags to pass to the C compiler

For each parameter, there may be several possible settings described in the *Makefile*. A leading `#` character is used to comment out every setting except one, which is the default setting. To select a different setting, put a `#` in front of the default and remove the leading `#` from the one you want to use.

If you're going to install *tcsh* somewhere other than the default location, you need to make two changes to the *Makefile*. Suppose you want to install *tcsh* as */bin/tcsh*. First, set the pathname that gets compiled into the *tcsh* binary. Find the `DFLAGS` line that you're using and add a definition for the `_PATH_TCSHELL` macro to it. If the `DFLAGS` line looks like this:

```
DFLAGS=
```

then change it to this (be sure to type the quotes as shown):

```
DFLAGS= -D_PATH_TCSHELL='"/bin/tcsh"'
```

(If `DFLAGS` has a non-empty value, just add `-D_PATH_TCSHELL='"/bin/tcsh"'` to the end of whatever's there already.)

Second, set the directory used by the installation commands when you install *tcsh*. Look for the line that sets the `DESTBIN` variable:

```
DESTBIN = $(TCSHTOP)/bin
```

Change that line to this:

```
DESTBIN = /bin
```

To change where the manual page is installed, set `DESTMAN` to the installation directory and `MANSECT` to the extension used for the file in that directory. To install *tcsh.man* as */usr/man/mann/tcsh.n*, the settings should look like this:

[1] If you modify either *Imakefile* or *imake.config* later, you'll need to rerun *xmkmf* and *make depend* to bring the *Makefile* and the dependencies up to date again.

```
DESTMAN = /usr/man/mann
MANSECT = n
```

Configure config.h

config.h contains some general system-dependent configuration flags used at compile time. It's created from one of the files in the *config* directory.

If you use *imake*, you don't need to create *config.h* yourself. The *Makefile* generated from the *Imakefile* includes a command to create *config.h* for you by selecting the proper file from the *config* directory.

If you're not using *imake*, look through in the *config* directory and determine which of the files there is most appropriate for your system. Then copy it into the main *tcsh* distribution directory as *config.h*. For example, the *hpux8* file works for both HP-UX 8.xx and 9.xx, so on my HP 715 running HP-UX 9.05 I do this:

```
% cp config/hpux8 config.h        Create config.h from vendor file
% chmod 644 config.h              Make config.h writable
```

Normally you won't need to modify *config.h*, but you should take a look through it just in case there are minor tweaks you think would be helpful. (If you do modify *config.h*, make a copy of it for reference because it gets removed if you run *make clean* later.)

Configure config_f.h

config_f.h contains several compilation flags that turn on or off various *tcsh* capabilities. Look through it to see whether or not you want to change any of them. For example, if you don't have *locale.h* on your system and can't compile in Native Language System (NLS) support, turn that feature off by changing this line:

```
#define NLS
```

to this:

```
#undef NLS
```

If you want the *tcsh* command editor to default to the *vi* key bindings instead of the *emacs* bindings, change this line:

```
#undef VIDEFAULT
```

to this:

```
#define VIDEFAULT
```

Compile tcsh

After you've edited the build files so they have the correct configuration parameters, generate *tcsh*:

```
% make
```

If the *make* command doesn't generate an executable *tcsh*, take a look at the last half of the *README* file to see if there are known workarounds for the problems that occur. Also, read the *FAQ* file and examine *Ported* to see if you overlooked any flags that are needed for building *tcsh* on your type of system.

If you're having *imake* problems, contact me at *dubois@primate.wisc.edu*.

Porting tcsh to a New System

If *tcsh* has not been compiled on the kind of system you have, you may not be able to find the appropriate configuration information for your machine. In that case, try to make educated guesses based on the parameter values from whichever systems are closest to yours. You may also want to add a new entry to *host.defs*, which describes how to set some system-related environment variables.

After you're done, please send your changes to *tcsh@mx.gw.com* so they can be incorporated into future releases.

Testing and Installing tcsh

After building *tcsh*, you should test it:

`% ./tcsh`	Start the *tcsh* you just built
`%` *...run some commands...*	See how it works
`% exit`	Terminate it

Point 8 of the *README* file suggests some special commands you can use to exercise *tcsh*. It's also good to try doing some everyday work with the newly built shell to see how it performs under ordinary circumstances.

When you're satisfied that *tcsh* is stable, install the binary and the manual page using these commands:

`% make install`	Install the *tcsh* binary
`% make install.man`	Install the manual page

If you encounter problems, try to determine the circumstances under which they occur. Consult the *README* and *FAQ* files, and verify by looking in *Ported* that you compiled *tcsh* with any special flags needed on your system.

Allowing tcsh To Be a Login Shell

After you've installed *tcsh*, you're almost finished. The final step is to make sure that *tcsh* can be used as a login shell. Typically you select a login shell using a command like *chsh* or *passwd −s*. These commands will likely require that *tcsh* be registered with the system as a trusted shell. Also, your FTP server may reject connections to accounts that have *tcsh* as the login shell unless *tcsh* is registered as a trusted shell.

The most common way to tell the system which shells are trusted is the */etc/shells* file. See if there is a *getusershell*(3) manual page to find out whether this is true for you. Typically, the trusted shell list is determined as follows:

- If */etc/shells* exists, the shells listed in it are considered trusted login shells. If */etc/shells* is already present on your system, you only need to add the *tcsh* pathname to it.
- If */etc/shells* doesn't exist, there is a set of shells that the system considers to be trusted login shells by default. (The *getusershell*(3) manual page should say what they are.) In order to register *tcsh*, you must create */etc/shells* and add the pathname for *tcsh* as well as the paths of all the default shells. If you put only *tcsh* in the file, then the default shells will no longer be considered trusted!

Entries in */etc/shells* must be full pathnames. Here's what it looks like on one of my systems:

```
/bin/sh
/bin/csh
/bin/ksh
/bin/tcsh
```

After modifying */etc/shells*, try changing your login shell to *tcsh* to verify that the system accepts it. You should also be able to use *ftp* to connect to your machine with your regular login name and password:

```
% ftp yourhost
Name: yourname
Password: yourpassword
```

If Your System Doesn't Use /etc/shells

Some systems use a file other than */etc/shells* to identify which shells are considered legal, and it may have a different format. For example, */etc/shells* is replaced under AIX by the `shell=` line in */etc/security/login.cfg*. Consult your local documentation for specifics.