The tcshrc manual

A guide for the tcshrc package

Simeon (Simos) Xenitellis

Amine Chelghaf

The tcshrc manual: A guide for the tcshrc package
by Simeon (Simos) Xenitellis and Amine Chelghaf

This document serves as an introduction to the tcsh shell, providing pointers to further documentation and covering significant features of the tcshrc package.

The latest version of this document can be found at the tcshrc WWW site at *http://tcshrc.sourceforge.net/*.

Revision History

Revision 0.5 8th Dec 2000 Revised by: simos at hellug dot gr
First public distribution
Revision 0.6 22nd Jan 2001 Revised by: simos at hellug dot gr
Documentation of the F* keys, addition of CSH/TCSH book, tcsh mailling list.
Revision 1.0 2nd Sep 2001 Revised by: simos at hellug dot gr
Added documentation for the installation, tcsh_config utility.

Table of Contents

Chapter 1. Shells

The shell is an interface to the operating system just like the graphic user interface (GUI) is. Typically, the shell acts as a command interpreter; it takes each command and passes it to the operating system. It then displays the results of this operation on your screen.

The most common shells are

- Bourne shell (sh)
- C shell (csh)
- Korn shell (ksh)
- TC Shell (tcsh)
- Bourne Again Shell (bash)

To identify which shell you are currently running, you may type in your current shell **`echo $shell`**

Historically, the shell came before the GUI, however the latter has become more popular since it allows in several cases easier usage of a computer. In reality, the shell is well-suited in specific types of usage and extra functionality that it may provide can make it quite productive.

This document describes the tcshrc project, a project that demonstrates exactly this extra functionality of the tcsh shell and makes it more accesible to the end user.

Chapter 2. Basic resources on Unix, shells and the tcsh shell

As basic reference, the reader is advised to read the following

- The Using csh & tcsh[1] handbook from O'Reilly is the definitive hard-copy manual for csh and tcsh. At the mentioned URL one can find some electronic versions of documentation of csh and tcsh.
- UnixIntro[2] is a gentle introduction to Unix. It has a chapter dedicated to shells and provides an HTML version of the tcsh man page.
- The Unix FAQ is available at the comp.answers[3] newsgroup. It is excellent to read, especially if you have tried to use Unix and have questions that you want answered.
- The Path HowTo[4] describes common tricks and problems with Unix or Linux environment variables and it focuses on the PATH variable.
- A further link that describes how to add search directories in the PATH variable is the Search Path Essentials link[5].
- The tcsh man pages is an invaluable source of information on the extra features of tcsh. It is recommended for the reader to walk through the sections of the man pages and use the features while reading.

There is a development maillling list on tcsh. To find more information on how to subscribe, send an e-mail to listserv@mx.gw.com[6] and put the word *help* in the message body.

The author and developer of tcsh is Christos Zoulas. He is available at the mentioned maillling list.

Notes

1. http://www.primate.wisc.edu/software/csh-tcsh-book/
2. http://www.mhpcc.edu/training/vitecbids/UnixIntro/UnixIntro.html
3. news:comp.answers
4. http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/Path.html
5. http://www.ecn.purdue.edu/ECN/Newsletters/1996.September/Search_Path_Essentials
6. mailto:listserv@mx.gw.com

Chapter 3. Installation of the tcshrc package

To install the tcshrc package for your current account, choose from the following scenarios.

I want a quick installation for my account only.

`make install` at your command prompt.

The above command essentially keeps a backup of the existing `.tcshrc`, if there is one, as `.tcshrc.ORIGINAL`. If such a backup file already exists, no backup is kept in order not to remove the initial .tcshrc file while doing repetitive installations.

To make a backup of your existing .tcshrc file

```
% cd
% cp .tcshrc .tcshrc.ORIGINAL
```

I want to make a system installation

To make a system installation, type

```
# make systeminstall
```

AND ALSO use the `tcsh_config` configuration utility with the -s (–system) option to populate `/etc/skel`.

This installs tcshrc in $(PREFIX)/share/tcshrc/ and adds the `tcsh_config` configuration utility in $(PREFIX)/bin. The `tcsh_config` utility can be used to make the installation in the `/etc/skel`. Finally, tell your users that they can add tcshrc to their existing accounts by running

```
% tcsh_config –user
```

Note: The `tcsh_config` configuration utility is usually placed in `/usr/local/bin` which is not in the path in default Unix installations. You can invoke the utility using the absolute path as in

```
% /usr/local/bin/tcsh_config –user
```

Chapter 4. tcshrc FAQ and features

For a full list of the tcshrc features of tcsh that are made available, you should consult the `.tcshrc.*` files and the tcsh man pages. Moreover, there must be a book on the subject from the favourite bookshop.

Q: I did "tar xvfz tcshrc-0.x.tar.gz" and I did not find any RC files!

A: The RC files start with a dot, which means that you need to do

```
% ls -al
```

to see them.

Q: When I try the completion, it does not work!

A: You are doing

```
%cp /etc/passwd /etc/pa<TAB>
```

The behaviour set in the .tcshrc files is to complete only in safe mode. This means that it is not sane to complete on existing files in this case.

Use

```
% cp /etc/passwd<ENTER>
cp: error blah blah
% <UP ARROW>
% cp /etc/passwd <F7>
% cp /etc/passwd /etc/passwd<ENTER>
% cp /etc/passwd /etc/passwd<.old>
```

The F7 key prints the last argument of the previous command.

Q: When I do "cd<TAB>", I only see directories. This is nice!

A: This is part of the *completion* functionality, and it is configured in `.tcshrc.complete`.

Q: Auto-correction confuses me.

A: Try this

```
% cd /usr/locl/bin<ENTER>
CORRECT> cd /usr/local/bin (y|n|e)?<SPACE> or <y>
%
```

For the opposite, suppose you really want to make the directory `/usr/locall`

```
% mkdir /usr/locall<ENTER>
CORRECT> mkdir /usr/local (y|n|e)?<ENTER> or <n>
%
```

Consider that the SPACE key means *acceptance of suggestion* while the ENTER key means *forcing what we wrote*.

Q: What does the F1 key do?

A: It prints the man page (if there is one) of the current command you are trying to execute. It is used like in the following.

```
% ncftp<F1>

the command "man ncftp" is executed. once you have found the information
you are interested in, you may hit q and continue
editing the command line.
% ncftp
```

It does not disturb the command you are current editing. At present, the *man* application is used to provide full help documentation. Another option is to use *whatis*. If you are aware of a utility that provides short help of up to 15 lines for typical UNIX commands, the author would be glad to hear about.

Q: What do the F2 and F3 keys do?

A: Suppose you are editing a long line. F2 sets a mark to the command line while F3 is used to switch between the current position and the marked position.

Q: What does the F4 key do?

A: Currently nothing.

Q: What do F5 and F6 do?

A: F5 does a spelling check on the current line while F6 spell-checks the current word the cursor is at. Spelling is typically done by checking that filenames or directories exist. You may find it usufull when you type a long command line without hitting the TAB key and you notice that some filenames are not correct.

```
% cd /usr/loca/bin<F5>
% cd /usr/local/bin
```

Q: What does the F7 key do?

A: Without F7, you would do

```
% cd /tmp
% ls demodir
not found blah blah
% mkdir demodir
% cd demodir
```

With F7, you do:

```
% cd /tmp
% mkdir demodir
% cd <F7>
% cd demodir<ENTER>
```

Try several times to get used to it. F7 prints the last argument of the previous command.

Q: What does F8 do?

A: F8 searches the command history for commands that match the current line, up to the cursor. For example, suppose we want to invoke the history line 18. In a real case, we do not need to type *history* to check out the previous commands. We typically remember them.

```
% history<ENTER>
  17 21:54  clear
  18 21:59 cd /usr/local/samba/bin
  19 21:59 ls -l
  20 22:00 cd ../lib
  21 22:02 vi *
  22 22:02 cd
  23 22:02 smbclient -L //SERVER/share -U test
  24 22:02 history

% cd<F8>
% cd
% cd <F8>
% cd ../lib
% cd ../lib  <F8>
% cd /usr/local/samba/bin
```

Q: What does F9 do?

A: It simply clear the current screen. It is smarter than a simple `clear` command, it does not affect the current line you are editing.

Q: What does F10 do?

A: `ls -l` is among the most common commands one typically types. F10 does just that, it executes that command. This one does affect the current line you are editing. A workaround is sought that does not affect the current line.

Q: What does F11 do?

A: It shows the current load of the system and like F9, it does not affect the current line you are editing. On the other hand it does not appear to be enabled on Linux. It remains there until a solution is found.

Q: What does F12 do?

A: It does standard completion, irrelevant of the smart and configurable completion that tcsh provides. There are some cases that the smart completion is not smart enough. This is where F12 comes in. For example, suppose you have a file called lynx_bookmarks.html and you want to make a backup adding an extension. You may use the trick described elsewhere in this FAQ, using the F7 key, or you can do the following.

```
% cp ly<TAB>
% cd lynx_bookmarks.html
% cd lynx_bookmarks.html ly<F12>
% cd lynx_bookmarks.html lynx_bookmarks.html.old<ENTER>
```

Q: What do these lines mean?

root has logged on pts/0 from :0.
root has logged on pts/1 from :0.
root has logged on tty1 from local.

A: They are part of the *watch* facility. Whenever you run a command in tcsh, tcsh checks if someones has logged on/off the system. If so, it will print it here. It is quite handy to know what is going on your system. It is even handy if you run a non-networked system, to know where you have shells open.

Q: I was compiling an application and at the end of it I got

Time spent in user mode (CPU seconds) : 8.810s
Time spent in kernel mode (CPU seconds) : 1.030s
Total time : 0:11.66s
CPU utilisation (percentage) : 84.3%
Times the process was swapped : 0
Times of major page faults : 29424
Times of minor page faults : 28521

A: These are some statistics that TCSH can provide (in a human readable manner) using a special command (time). With the current configuration, it is printed automatically whenever a process takes quite a bit of time to complete. It shows the time the process spent in kernel and user mode, the total time used for the process, the CPU utilisation (user+kernel time / total time) in %. The swapped times is the times the whole process was swapped. If you have plenty of memory, you usually get 0 here.

For the page faults, a small operating system tutorial. In modern operating systems, memory is used in chunks called pages. These pages can be swapped to the swap partition to make space for other processes. When our process is running and it cannot find one of its *pages*, it issues a *fault*, or a *page fault* and makes arrangements to have the page up. The fewer the page faults, the better. The *page fault* terminology is a bit wierd and it comes from long time ago.

> **Note:** the amount of page faults look to me rather a lot, perhaps they are faults regarding the presense of the page in the memory cache and not the swap. If a kernel hacker knows about this stuff and can have a look in the source code of TCSH, please clarify this issue.

Q: I find it a bit awkward when I change back and forth directories. Any nice trick?

A: You can use the *cd -* command. It takes you to the previous directory. Try it once more to take you to the initial directory.

Q: How can I access that *cd /usr/local/samba/lib/* command I type ages ago?

A: As long as you remember the beginning of the command that resides in the history, you can access it quickly.

Do

```
% cd /usr/local/samba/lib/
% cd /usr/local/bin/
```

```
% cd /etc/
```

Now you want to go to samba/lib.

```
% cd <ESC p>
% cd /etc<ESC p>
% cd /usr/local/bin<ESC p>
% cd /usr/local/samba/lib
```

That is, you press several times ESC p to go to the Previous occurence of a similar command. If at some point you want to go to the next command in the history, hit ESC n.

Q: What does ^G mean?

A: Means you press Ctrl-G. Speaking of ^G, try

```
% echo <^V><^G><ENTER>
```

You will hear a BEEP. ^V is used to *mask* the next character pressed.

Q: I want F4 to beep to me!

A: You need to bind F4 with the beep. Beep is ^G. Do

```
%bindkey -c <^V><F4> "echo -n <^V><^G>"<ENTER>
```

That's it. You need to use it on an empty command line.

Q: How can I traverse the history (backwards and forwards) restricting the search to what I have typed until now?

A: This is similar to the F8 key that goes only backwards. Check the question on the F8 key.

To go backwards, use the **ESC p** combination. To go forwards, use the **ESC n** combination. p is for previous and n for next. You hit first ESC, then you hit either p or n.

Q: How can I make it so that these .tcshrc* files are added to each newly created user?

A: You need to copy them to the `/etc/skel` directory. Then, the utility that creates new users will automatically include the tcsh configuration files. We assume that the tcsh shell has been chosen for the new user. In the future a script will be written that installs these files on a per-user basis.