Hentzenwerke Whitepaper Series

The Top Ten Command Window Tips

By Whil Hentzen

Power users can go a long way just using the graphical user interface (GUI) running on top of Linux. You can use the GUI for just about everything, from making changes to the system configuration to launching programs. However, the GUI is a fairly recent addition to the Linux world. Many hard-core and experienced users never use the GUI, and there are uses for Linux for which a GUI would be out of place. This GUI-less world sports an interface that can be entered by using a tool similar to the Windows command window or DOS prompt. In this whitepaper, I'll discuss how to get under the hood by using the Linux command window.

Hentzenwerke Publishing, Inc. • books@hentzenwerke.com • www.hentzenwerke.com

1. Preface

1.1 Copyright

1.2 Revisions

1.2.1 History

Version	Date	Synopsis	Author
1.0.0	2004/5/10	Original	WH

1.2.2 New version

The newest version of this document will be found at www.hentzenwerke.com.

1.2.3 Feedback and corrections

If you have questions, comments, or corrections about this document, please feel free to email me at 'books@hentzenwerke.com'. I also welcome suggestions for passages you find unclear.

1.3 References and acknowledgments

Thanks to MLUG members Aaron Schrab and Glenn Holmer, among many others.

1.4 Disclaimer

No warranty! This material is provided as is, with no warranty of fitness for any particular purpose. Use the concepts, examples and other content at your own risk. There may be errors and inaccuracies that in some configurations may be damaging to your system. The author(s) disavows all liability for the contents of this document.

Before making any changes to your system, ensure that you have backups and other resources to restore the system to its state before making those changes.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

1.5 Prerequisites

This document was written using Fedora Core 1.0 and assumes a beginner's familiarity with use of Linux via the GUI and the Command Window.

2. The One Billion Names for the Command Window

The Linux command window is known by a variety of names; go to a Linux user-group meeting and you'll hear terms such as "command line," "command prompt," "shell" or "shell prompt," "terminal window," "command line," and "console," and perhaps even a misplaced "DOS prompt" or two.

Strictly speaking, "Command Line Interface" (CLI) is the generic term for the software device used to issue commands to the system. CLIs are called by different names depending on what operating system you're using, and how you're accessing the system. For example, Unix systems refer to them as "shells." If you have direct physical access to the machine, "console" is proper usage. The application on Fedora Core and on the Macintosh OS X operating system is called "Terminal," so common terminology refers to "opening another termiCron definition - what is it and what does it do?nal" or "opening another terminal window."

2.1. The many shells of Linux

Because we're using Linux, you'll run into the term "shell" rather often. "Shell" is a generic term that refers to a class of programs that provide CLI functionality. Most Linux distributions come with a number of shell programs, including the Korn Shell (ksh), the C Shell (csd) and the Bourne Again Shell (bash). In Fedora Core, the default shell program is bash; to get to it, select Main Menu | System Tools | Terminal.

To find all available shells on your system, type:

```
cat /etc/shells
```

Each of these shells performs the same functions, but they understand different commands and provide different built-in functions—consider OpenOffice.org Calc, Microsoft Excel and Lotus 1-2-3, which are all spreadsheets, but all of which have different menu structures and built-in spreadsheet functions.

By contrast, Windows has only one shell, called "command.com," and accessed through Program Files | Accessories | Command Prompt. You can find oodles of additional information on shells here: **http://www.faqs.org/faqs/unix-faq/shell/shell-differences/**

Getting back to the definition discussion - common terminology, then, often refers to "opening a shell."

When you open a CLI such as bash in Linux, you're greeted with a window that contains an operating system prompt, like so:

```
[whil@freedom whil]$
```

You type a command, press Enter, and the system (hopefully) responds to the command with some sort of output, or perhaps another shell prompt. For example, if I entered the command "ls -al" to list the files in the current directory, I might see something like:

```
[whil@freedom whil]$ ls -al
drwx------   58 whil     whil         8192 Apr  1 18:19 .
drwxr-xr-x    8 root     root         4096 Mar 31 13:48 ..
drwxr-xr-x    3 whil     whil         4096 Mar 23 14:16 Desktop
drwx------    8 whil     whil         4096 Mar  2 20:36 evolution
-rwxr-xr-x    1 whil     whil      1324820 Mar  7 18:27 installation.pdf
drwx------    2 whil     whil         4096 Feb 25 18:01 mail
drwxr-xr-x    1 whil     root         4096 Mar 19 12:08 music
drwxrwxr-x    2 whil     whil         4096 Mar 22 16:34 RollerStuff
drwxr-xr-x    4 whil     whil         4096 Mar 30 16:32 zips
[whil@freedom whil]$
```

Not all commands return a response; a lot of Linux commands produce no output if they succeed, so all you'll see is another prompt.

The operating system prompt is often referred to as a "command line" or a "command prompt." Programs that are designed to run by typing a command at the operating system prompt are often called "command line utilities."

The command window is a rich, powerful, programmable tool, and many books have been written that cover it alone. Here are the top 10 things about the command window I've found to be useful. The following discussion assumes that you're running bash as your shell program.

3. The Command Window title bar

The string in the title bar defaults to the name of the current user concatenated with the host name (machine name) and the current directory. If you open a command window and change to a different directory, like so:

```
cd /home
```

The string in the title bar will change.

4. The Command prompt

The string in front is the prompt. The default bash prompt includes the user, the host name, and the current directory. Typically, your command window will start in your home directory, so

```
[whil@freedom whil]$
```

means

```
[whil@freedom /home/whil]$
```

If you're in the “zips” subdirectory of your home directory (i.e. /home/whil/zips), you'd see

```
[whil@freedom zips]$
```

5. Command prompts for root vs. regular user

The prompt ends in either a dollar sign ($) or a pound sign (#). The $ means that you are logged on as a regular user. The # means you’re logged on as root. You'll hear this again, but note that if you're root in a command window, you can easily mangle or destroy your system (and possibly blow up the world) with one typographic error.

6. Customize your prompt

You can customize the prompt by making changes to certain configuration files.

6.1. Shell configuration files

The behavior of the command window (more specifically, the behavior of the shell) depends on values in several configuration files. Linux looks at the /etc/profile file first, then the ~/.bash_profile file, and then the ~/.bashrc file. (If you're using a different shell, these files are different.)

The first file is a generic, system-wide file that runs regardless of which user is logging on, and only the system administrator can modify this file. As you can see, there is a copy of the second file in every user’s home directory, and it runs every time that user logs in. Because each user has her own ~/.bash_profile file, the command window can be customized on a per-user basis.

The third file, which also exists in every user’s home directory, also runs on a user-by-user basis whenever a new command window is opened, except during initial login. As a result of having separate .bash_profile and .bashrc files, each user can run specific customizations just during initial login or for all command windows except during initial login.

You might be thinking that while this complex mechanism makes for wonderful flexibility, it’s also awkward if you want to run some customizations for a specific user, but also for every command window, because you’d have to add the same customizations to both files. Ha! Gotcha covered here too. Fedora Core’s default .bash_profile file comes with a short script that looks like this:

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
   . ~/.bashrc
fi
```

Don’t worry about how it works; the important thing is that this script makes bash run the .bashrc file in the user’s home directory whenever the .bash_profile script runs. As a result, if you put all of your customizations in .bashrc, they’ll run for every shell. If you still need customizations that run only during login, put them before this line in .bash_profile.

It's a good idea to put anything that takes a noticeable amount of time in .bash_profile so that sub-shells start up really fast by doing only the minimum in .bashrc.

And, of course, this all applies to the root user as well as every other user. The root user’s .bash_profile and .bashrc files are in /root, of course.

6.2. Customize your prompt string

Now on to the business of customizing the prompt, since that’s what we started talking about. The commands I’ll be discussing all go into the ~/.bashrc file, right? The basic structure of the command to change the prompt looks like this (note that there can’t be spaces on either side of the Equals sign):

```
PS1='Hello world!'
```

Enter this string into your .bashrc file, save the file, and then open a command window. The command prompt will simply say:

```
Hello World!
```

That’s probably not going to be very helpful, although I’ve seen a number of whimsical variations, the best being:

```
Yes, Master?
```

You'll notice that you have to enclose a string that has a space in it with quotation marks, which is probably no big surprise. If you've got a string that already has a quotation mark in it, you'll need to use a different delimiter, like so:

```
PS1="Whil's computer awaits direction:"
```

There are a number of special characters that will display certain types of information, such as the name of the user currently logged in, the name of the computer, the current directory, the date, the time, and so on.**Table 1** lists some common ones.

***Table 1**. Special characters you can use to customize your prompt.*

Special Character	Text Output
\a	An ASCII bell character (07)
\d	The date in "Weekday Month Date" format (for example, "Tue May 26")
\e	An ASCII escape character (033)
\h	The host name up to the first '.'
\H	The host name
\n	Newline
\r	Carriage return
\s	The name of the shell
\t	The current time in 24-hour HH:MM:SS format
\T	The current time in 12-hour HH:MM:SS format
\@	The current time in 12-hour am/pm format
\u	The user name of the current user
\v	The version of bash (for example, 2.00)
\w	The full name of the current working directory (except for your own home directory, which will be displayed as '~')
\W	The basename of the current working directory (for example, 'cups' instead of '/etc/cups' and 'whil' instead of '~')
\!	The history number of this command
\#	The command number of this command
\$	If the effective UID is 0, a #, otherwise a $
\\	A backslash
\[	Begin a sequence of nonprinting characters, which could be used to embed a terminal control sequence into the prompt
\]	End a sequence of nonprinting characters

Typing "man bash" at the command prompt () will give you a full list of special characters.

Table 2 contains a couple of examples to get you started. Note that you can just type the command in a command prompt; you don't have to modify your .bashrc file and save it each time.

***Table 2**. Some sample prompt changing commands and their results.*

This command	Produces these results

`PS1=[\w]`	`[~]`
	(when you're in your home directory)
	
	`[/etc/cups/interfaces]`
	(when you're in the /etc/cups/interfaces directory)

`PS1='[\w] Dude? '`	`[~] Dude?` (when you're in your home directory)
`PS1='[\u@\h \W] Dude? '`	`[whil@freedom cups] Dude?` (when you're in the /etc/cups directory)
`PS1='[\t] [\u@\h:\w]\$ '`	`[21:52:01] [whil@freedom:~]$`

6.3. Customize your prompt colors

Now that you've had fun customizing the text of the prompt, let's go another step and customize the color of the prompt. Why would you want to? Many experienced Linux users change the color of the prompt for their root account so that it's very, very clear when they're logged in as or have "su'd" to root.

In order to change your prompt's colors, you may have to specify both the foreground (the characters themselves) and the background (the color of the screen behind the characters) colors. If you're not careful, you may end up with an unreadable combination, like yellow on white or magenta on black.

You may want to issue these commands in a command window while experimenting – if you end up with an unreadable prompt, all you need to do is close the window and open it up again. What's really handy is that not only are you back with your original prompt, but your command history is saved, so you'll be able to recall old experimental commands instead of retyping all of those arcane characters.

In order to change the color of the foreground or background, you include in your prompt string a series of characters like so:

```
PS1="\[\e[34mX"
```

What do all these characters mean? The '\[\e[' string is an escape sequence that for our purposes here we won't worry about (one thing it does is tell the computer not to print certain characters). The 34 changes the foreground color to blue (a table listing numbers that map to colors follows shortly), and the 'm' terminates the escape sequence. And the 'X' is our temporary prompt, which we'll change back to our original customized prompt once we're done experimenting with colors.

Now, about that numbers and color table. See Table 3.

***Table 3**. Prompt string color values.*

Value	Effect
30	Make foreground (the characters) black
31	Make foreground red
32	Make foreground green
33	Make foreground yellow
34	Make foreground blue
35	Make foreground magenta
36	Make foreground cyan
37	Make foreground white
40	Make background (around the characters) black
41	Make background red
42	Make background green
43	Make background yellow
44	Make background blue
45	Make background magenta
46	Make background cyan
47	Make background white (you may need 0 instead, or in addition)

Note that there is no space between the X and the characters you type, and that the characters you type will be the same color as the prompt string.

In order to provide room between the prompt string and the commands you type, include a space after the X, like so:

```
PS1="\[\e[31mX "
```

And if you want the text you type to be a different color, say, black, set that after the prompt, like so:

```
PS1="\[\e[34mX\[\e[30m "
```

Now, once you've got the hang of the colors, include the rest of the prompt that you so painstakingly constructed:

```
PS1="\[\e[34m[\u@\h \w] Dude? \[\e[30m"
```

And you're all set with a customized prompt string that very well may be a horse of a different color.

7. Access root inside the Command Window

7.1. Change to root inside the command window

You can change to root after opening a command window in one of two ways. The first is by using the "su" command:

```
[whil@freedom whil]$ su
Password:
[root@freedom whil]#
```

Note that you're still in the /home/whil directory, but now you're root. (Referring back to the first item, you'll note that the title bar has also changed, which means you have "godlike" powers. You can make any change you want to the system, install software, delete files, get into anyone else's home directory, and so on. This also means, as I've said before, that you can do tons of damage. For example, "rm -rf /" will destroy every file on your computer in about five blimptoseconds.

The second way you can change to root is by using the "su -" command:

```
[whil@freedom whil]$ su -
Password:
[root@freedom root] #
```

If you use this method, you'll note that both the prompt string and the title bar have changed. Also note that you're now in the /root directory (which is the root user's home directory). The reason that this is different is that the "-" after the "su" command brings root's environment along with it. What's an environment? It's a group of settings, such as the prompt, system variables, and so on, that belong to a specific user. The home directory, for example, is one such setting, and the command-prompt string is another. These can be different (and often are) for different users.

When you use the "su" command, you're just gaining root privileges, but you're still working in the original user's environment—with the original user's path, prompt, home directory, and so on.

When you use the "su -" command, you switch your environment to that of root, and that's why you're suddenly launched into root's home directory. "su -" is short for "su -l" or "su --login", by the way.

7.2. Change from root back to the regular user in the command window

Once you're done with your root business, you could close the Command Window and start up a new instance that would feature the regular logged in user again, but that would be time-consuming and unnecessary. Instead, in order to go back to the regular user from root, simply issue the "exit" command.

Of course, that might be too much work for you. You can also type ctrl-d, which enters the 'exit' command and presses Enter for you.

8. Commands are sensitive – case-sensitive, that is

One command is "ls" (list). Type "ls" at the command prompt, and you'll get a directory listing. Type "LS" (or "lS" or "Ls") and you'll get a response like this: "-bash: LS: command not found."

9. Move through command history

You can scroll through command history by using the up and down arrow keys. You can also type "history n" where 'n' is a number to display the last 'n' commands. Since bash stores the last 1000 commands by default, you may want to include a number that is fairly small (perhaps less than the number of rows in your command window).

```
history 10
```

Once the history file gets too big for you to comfortably navigate through it,

```
history -c
```

will clear the history list by deleting all entries.

10. String completion

Pressing the Tab key will complete strings. As you saw in Chapter 13, you can use the Tab key to complete a long file name instead of typing it (and creating typos). To try this out if you didn't go through Chapter 13, switch to a directory that contains a bunch of long file names. If you created a "zips" directory under your home directory in order to save zips and rpms that you've downloaded, your zips directory might look something like this:

```
fp-linux-ws.rpm
mozilla-i686-pc-linux-gnu-1.6-installer.tar.gz
rh9.ymessenger-1.0.4-1.i386.rpm
rp8.linux20.libc6.i386.cs2.rpm
rv9_libc6_i386_cs2.tgz
VMware-workstation-4.0.5-6030.i386.rpm
xmms-mikmod-1.2.8-1.i386.rpm
xmms-mp3-1.2.8-3.2.fr.i386.rpm
```

Suppose you wanted to run the "rpm" command on the VMWare rpm package instead of doing so through a graphical file manager. According to the VMWare documentation, the command would look something like this:

```
rpm -Uhv VMware-<xxx>.rpm
```

where the <xxx> is the string "workstation-4.0.5-6030.i386". You certainly don't want to type all of those numbers and periods and hyphens. Instead, you can type this:

```
rpm -Uhv VM
```

and then press the Tab key. The string will be expanded to this:

```
rpm -Uhv VMware-workstation-4.0.5-6030.i386.rpm
```

and the cursor will be sitting at the end of the line, waiting for you to make additional edits; or you can simply press the Enter key to execute the command.

The only caveat is that you have to type enough of the string that the shell will be able to uniquely identify the file you're interested in. If you didn't type enough to identify the file, the amount that is matched will display, and then you can continue typing until you get enough to make another unique match.

11. Multiple command windows

You can have multiple command windows open at the same time. This is handy for several reasons. The first is that some commands take over the command window, disabling it for any other use until the command is finished. For example, if you want to edit a configuration file, you can call gedit from within a command window, like so:

```
gedit /etc/fstab
```

However, as long as gedit is running, you don't have access to the command window that spawned the editor session. If you want to run another command, you need to open a second command window.

The second reason to open multiple command windows is so you can do something as root without logging in as root. If you've already got one command window open as your day-to-day user, you don't have to close it. Just open a second command window and change to root in that window.

12. Command window attributes

You can change the attributes of a command window. The collection of a command window's attributes make up its "profile," and you can create multiple profiles, each of which is used by a different user. This is particularly useful when you need to run one command window as root. Sure, the prompt changes, but it's easy (particularly as your eyes pass a certain vintage, like mine) to miss that small visual clue. (That's why I also change my root command prompt to be radically different from all others.)

To create a new profile for a command window, select File | New Profile (see **Figure 1**).

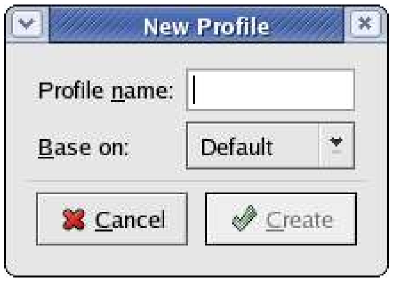


***Figure 1**. The New Profile dialog allows you to create a new profile, and optionally base it on an existing profile.*

You can also select Edit | Profiles to open the Edit Profiles dialog (**Figure 2**), and then click the New button to open the New Profile dialog as shown in Figure 1.

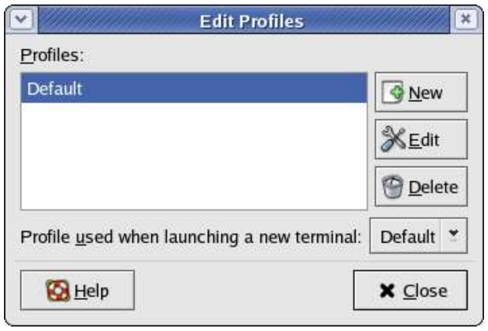
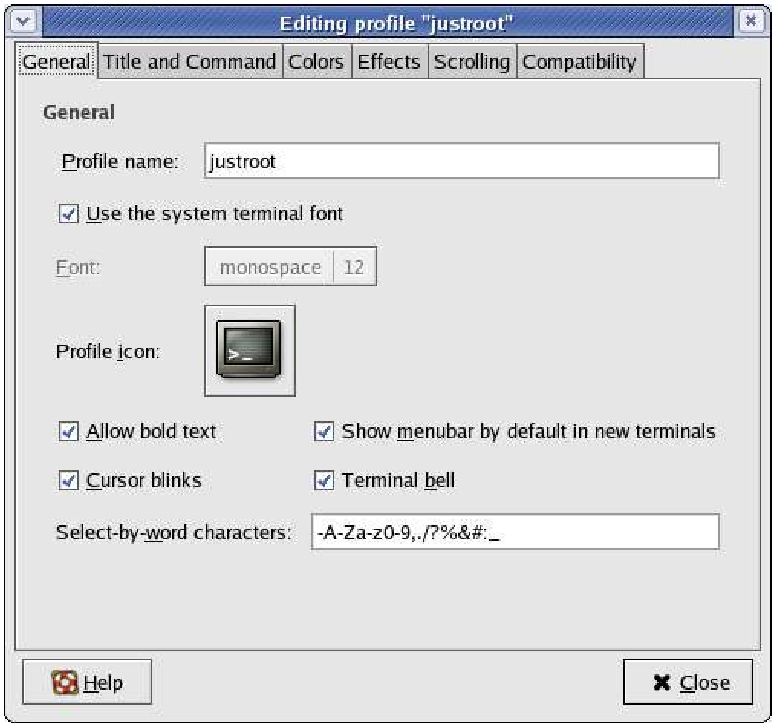


***Figure 2**. The Edit Profiles dialog allows you to manage multiple profiles.*

The New Profile dialog defaults to creating a new profile based on the default profile in the system, but if you've already created different profiles, you'll be able to create a new profile based on one of them as well.

Ultimately, the New Profile dialog doesn't do anything terribly interesting. It's the Editing Profile dialog, shown in **Figure 3**, that holds all the action.



General

Profile name: justroot

[✓] Use the system terminal font

Font: monospace 12

Profile icon:

[✓] Allow bold text [✓] Show menubar by default in new terminals

[✓] Cursor blinks [✓] Terminal bell

Select-by-word characters: -A-Za-z0-9,./?%&#:_

Figure 3*. The Editing Profile dialog contains six tabs of various attributes that describe the profile being edited.*

You can also get to the Editing Profile dialog by selecting Edit | Current Profile, or Edit | Edit Profiles, and then selecting the profile of interest and clicking the Edit button. Or you can right-click in the command window and select Edit Current Profile from the resulting context menu.

On the General tab, the profile icon is the image that displays in the panel when you have a command window running. You probably don't want to change the font, since the default choice, monospace 12, looks okay and renders fixed-pitch strings nicely. Changing to a proportional font such as Arial can make it nearly impossible to distinguish between certain characters (the lowercase letter "l" and a number "1", for example).

On the Title and Command tab, you can control what shows up in the title bar. Choices include the name of the profile in various combinations, with the default string of "username, hostname, and current directory."

13. Where to go for more information

This free whitepaper is published and distributed by Hentzenwerke Publishing, Inc. We have the largest lists of "Moving to Linux", OpenOffice.org, and Visual FoxPro books on the planet.

We also have oodles of free whitepapers on our website and more are being added regularly. Our Preferred Customer mailing list gets bi-monthly announcements of new whitepapers (and gets discounts on our books, first crack at special deals, and other stuff as we think of it.)

Click on "Your Account" at www.hentzenwerke.com to get on our Preferred Customer list.