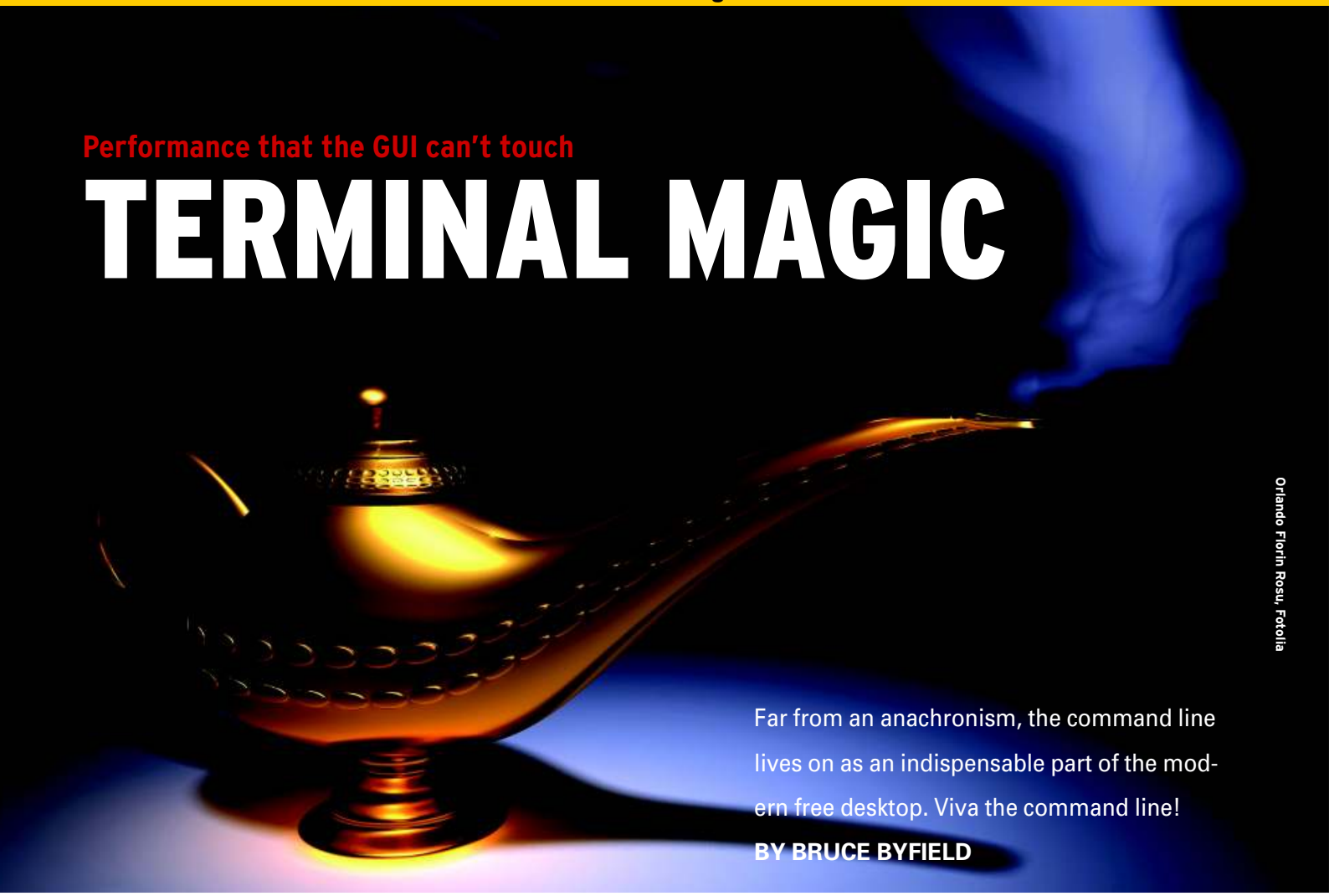
Performance that the GUI can't touch

TERMINAL MAGIC



Orlando Florin Rosu, Fotolia

Far from an anachronism, the command line lives on as an indispensable part of the modern free desktop. Viva the command line!

BY BRUCE BYFIELD

Does the command line still matter in GNU/Linux? Considering recent developments, you could be forgiven for wondering. Canonical is concentrating on making Gnome rival OS X, but no equally large effort is being made to improve the shell. A few ingenious applications such as xclip or terminator exist, but for every new shell program on the CLI-Apps site, there are dozens on openDesktop.org. These days, the wisdom is that the command line is for aging curmudgeons, and the future of free operating systems is on the desktop. Yet, despite a few limitations, the truth is that the command line still has distinct advantages over the desktop and probably always will.

I wanted to write this column because I believe in the value of the command line. In future columns, I'll channel the spirit of my late colleague Joe Barr, whose CLI Magic column ran for years, and I'll explore the tools and tricks that make the command line an indispensable part of a modern free desktop – and not the anachronism that the desktop-fixated often assume. A decade ago, no one needed to argue about this because the widespread opinion among GNU/Linux users was that the desktop worked fine for trivial purposes, provided that you used a minimalist window manager such as FVWM and avoided Gnome or KDE, which were already being described as "bloated."

Serious work, whether development, package installation, or any other administrative task, was done from the command line. If you hadn't compiled your own kernel, you were a dilettante and probably a Windows user as well.

When free desktops lagged behind proprietary ones, this command-line machismo had a certain justification, and you can still find it today in certain circles. The addition of new, non-technical users has made those who prefer the command line on their own operating systems a minority. Moreover, instead of catching up to the desktops of other operating systems, Gnome, KDE, Xfce, and other desktops have matured enough to set the standards for functionality and customization (if not also appearance or user-friendliness).

Just as importantly, many modern GNU/Linux users are refugees from Windows, where the command line has few of the conveniences of the Bash shell or any of its alternatives. If they have tried it, users are turned off by the awkwardness of the choices available to them.

When you start to think in terms of the best tools for the job, you soon realize that both the command line and the desktop have their advantages. Although the desktop is easy to learn and best suited for graphics work, the command line has the advantages of being consistent between distributions and of having less overhead, greater efficiency, and a more complete set of tools than desktop applications offer. These advantages are worth considering in routine work, but become especially valuable when you need to troubleshoot.

What the Command Line Doesn't Do

While appreciating the usefulness of the command line, you also must recognize its limitations. One such limitation is the time it takes to learn how to work on the command line (Figure 1). If you sit someone down who has used a desktop on another operating system, you can

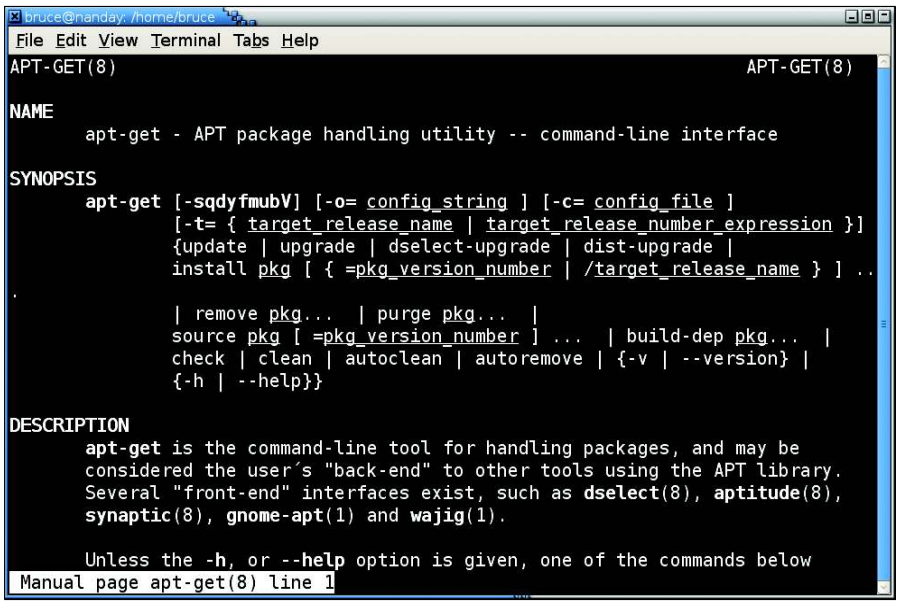


Figure 1: As apt-get (the basic package management tool) illustrates, shell commands can look overwhelming at first.

teach them the GUI basics in a few moments. Clicking an icon to start an application, opening a menu, finding open windows on a task bar, exiting a system – none of these tasks are particularly challenging.

The command line, though, is another story. Despite features such as tab completion and history, becoming comfortable on the command line requires more effort and perhaps even a crib sheet taped near your monitor. In return, you gain greater control over your computer, just as you gain greater efficiency when you learn to touch type rather than peck and hope.

Another limitation of the command line is that it works poorly with graphics, a consequence of trying to work with visual elements in a text-based environment. I used to say that one magazine was so old school that its editors considered vi a Photoshop replacement – the joke being that, when trying to manipulate graphics from the command line, you can only visualize and hope for the best. For example, you can use gPhoto2 instead of digiKam, but without thumbnails, you can only guess which pictures you want to download. In the same way, you can use Links or Lynx to browse the Internet, but even on sites where the designer has thought about text-only browsing, you can still miss a lot of essential content. Although you can design a slide show from the command line with tools such as latex-beamer or write HTML in a text editor, eventually you must revert to the desktop to check your work in the environment in which it will be used.

What the Command Line Does Well

So what does the command line do well? First, it offers a common interface among distributions that the desktop does not. Move away from Fedora, and you might miss its Network Manager or PackageKit. Move away from openSUSE, and you might miss the YaST configuration center. Move between Xfce and KDE in the same distribution, and it takes a while to feel comfortable. By contrast, no matter what distribution you use, you always have the Bash shell or something similar, so you know the basics of how to use it. The greatest difference you are likely to encounter is a pre-set alias or two that you can easily duplicate. For this reason, you might argue that the shell is one of the elements that guards against fragmentation on the desktop.

Another minor advantage of the command line is that it uses fewer resources than even the leanest desktop does. Any desktop requires a running X Window System, as well as the desktop itself, which adds considerably to the hard drive space and RAM that it requires, especially considering that many desktop applications are only front ends for command-line programs that run in the background. Because new systems are loaded with 3 to 4MB of RAM and terabyte hard drives, these requirements are less of an issue now, but many working systems still have half these specifications or less.

Try doing a complete backup of your home directory to an external hard drive with 1MB of RAM from the desktop, and if yours is anywhere near the size of mine, you are going to have a lot of hard drive churn as Gnome or KDE struggles to update its display. You might very well get a message that your file manager is not responding, leaving you to wonder whether it's only the file manager that has choked or whether the transfer has as well. By contrast, use *cp -rv* for the same operation, and you not only get little churn and no stalls, but you also know exactly how the copying is progressing.

The older your system, the greater the advantage. An unmodified four-year-old machine might be almost useless for running the latest Gnome or KDE, but you can still get extra service out of it as a

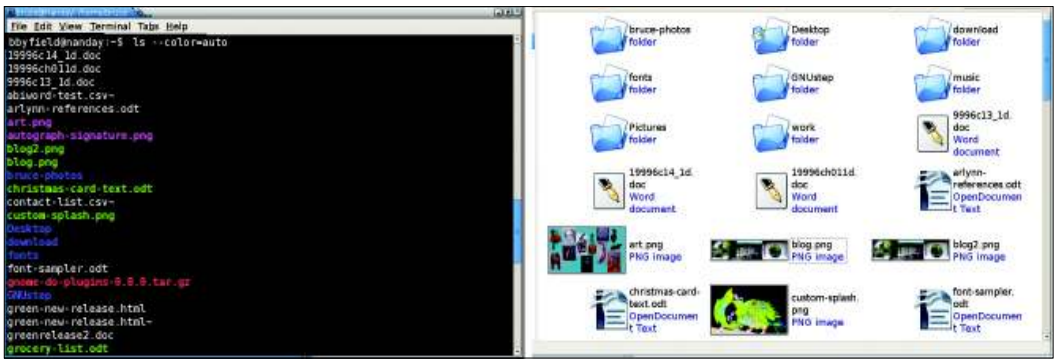


Figure 2: The cp command's use of color substitutes very well for the different icons that represent different types of files on the desktop.

firewall if you boot it using Bash. Instead of consigning it to a landfill, you can reuse the machine for another few years and pride yourself on adding a little greenness to your computing.

Another reason for favoring the command line is that, like keyboard shortcuts, shell commands are faster (once you know them) than selecting menu items or toolbar icons with the mouse. Using the command line to do that same backup of my home directory on my Debian machine to an external drive, I only need to enter *cp -rv /home/bbyfield/* /media/usb* – 31 characters that even someone who doesn't touch type should be able to enter in less than 10 seconds. Most tweets are longer (Figure 2).

To do the same operation in Gnome's Nautilus file manager, I begin by navigating to my home directory. Once there, I select the files and directories by pressing Ctrl + A or selecting *Edit | Select All*. To prepare to copy, I press Ctrl + C and either open the target directory in another Nautilus window or navigate to it and then press Ctrl + V or right-click and select *Paste* from the context menu. That's five steps to the command line's one, even when you streamline the process by using the keyboard as much as possible. Moreover, you need to locate the source and target directories, whereas with the *cp* command, your working directory doesn't matter. If you are in the source directory, your command becomes even shorter because you can replace */home/bbyfield* with *./*. Working with Nautilus takes at least three or four times longer than working from the command line, and even the use of a different file manager, such as Dolphin or Thunar, will take about the same amount of time. Randomly choose an administrative task and you will probably find that it is faster from the command line.

Another strength of the shell is that its commands inevitably include more options than their desktop counterparts. In theory, I see no reason why this observation should be true, but in practice, interface designers assume that desktop apps cannot have as many options as apps running from the command line. Perhaps the assumption is a watered-down version of the Pareto Principle – 80 percent of users' needs are satisfied by 20 percent of the functionality.

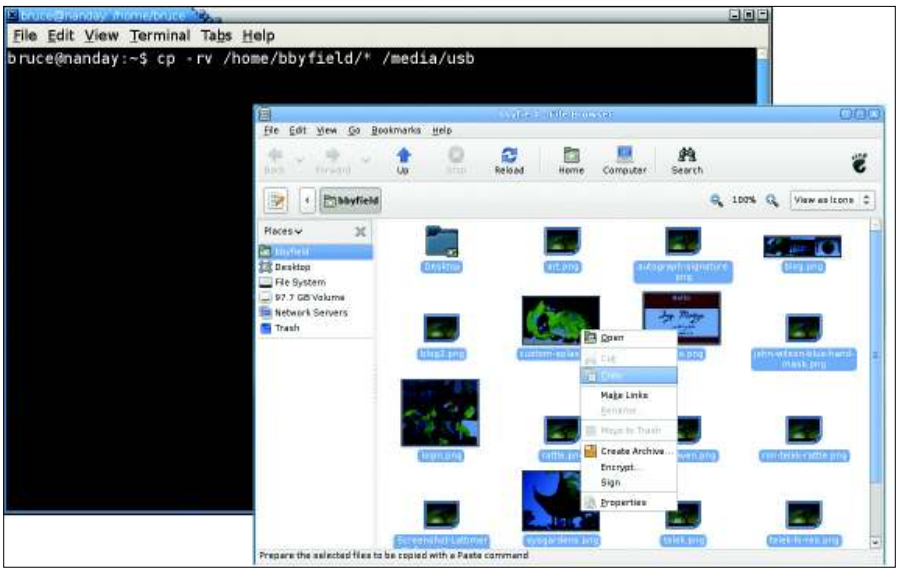


Figure 3: Copying from the command line takes only a handful of characters; however, from a file manager such as Nautilus, copying requires many more steps.

Consider, for example, the listing of a directory's content. On one hand, the only feature that Nautilus has that the *ls* command does not is the use of emblems to categorize files, and that can be duplicated by setting up a consistent file-naming system or a series of sub-directories. Even the use of different icons is easily duplicated with the option *--color=always* or *--classify* (Figure 3).

On the other hand, *ls* has numerous advantages over Nautilus. In Nautilus, you have to open the tree view to see both a directory and its contents, instead of entering *ls -R* from the command line. The *ls* command also has numerous sorting options that Nautilus lacks, including block size, inode, reverse order, file size, version, modification time, and SELinux security context. Even a special *--sort* flag exists to streamline the inputting of some of these options and add sorting by a designated string.

Use of the *cp* command instead of copying with Nautilus is also a nice perk. Although Nautilus is adequate for routine copying, *cp* has flags for creating a backup of copied files (*--backup*), creating symbolic links (*-l*), preserving file attributes (*--preserve=ATTRIBUTE*), and copying only newer files – and these are just those I've found most useful in my home administration.

If all these reasons for using the command line are not enough to persuade you in the abstract, consider this very practical and inescapable fact: If anything goes wrong with your system, you either need to use a recovery disk or boot in single-user mode. You might not have the X Window System working, and in any case, working without it simplifies your investigation. In this situation, you will be glad that you know the command line in all its low overhead, speedy, full-featured, graphicless glory.

Conclusion

One advantage of the command line that I did not argue is that it acts as protection against intrusions or accidents. You can see this argument applied in the settings for GDM, in which you have the option to prohibit desktop logins for remote users or root. The logic of this option is that if users do not have their familiar graphical tools, it's assumed that they are less likely to crack the system or bring it down through ignorance. But such security through ignorance seems only a variation of the discredited idea of security by obscurity, which gives a false sense of protection while offering no assurances at all. Besides, the reasons I have listed are more than enough to ensure that the command line will continue to be a valuable interface for computers for years to come.

Programs such as Gnome Do or KRunner could easily evolve to offer more command-line features, resulting in an interface that combines the best of the shell and the desktop, but even if such improvements do not occur, understanding the command line is still worth the effort. ■