# A simple 'yes' will do

*by David D. Scribner*

If you've used interactive commands in Linux, you know they can be both a blessing and a curse. Some commands are interactive by choice (like `cp`, `mv` and `rm`), but others may not even give you that option. Using these commands while at the command line usually present no problem, as you are there at the console to supply any needed input. However, put these commands in a shell script or cron job and you might be wondering just how you will be able to provide those answers while the script or cron job is running.

The answer just might lie with '`yes`'; a command included in with your Linux distribution as part of the core GNU (and *NIX) utilities. The '`yes`' program simply sends an endless string of "`y`" characters, followed by a newline (`\n`, the equivalent of pressing the `Enter` key) to STDOUT. Check it out for yourself and type '`yes`' at the shell prompt. That column of "`y`'s" you see on the left edge of your screen or terminal window is actually a stream of those characters bleeding down as fast as your processor can issue them. You'll have to break the stream with `Ctrl-C`, which will bring you back to your prompt. Not very useful by itself, but it works like a charm when you pipe the output to a command that can use it, such as those that operate in interactive mode.

So how can you use this to your benefit? For this example I'll use another command we all have at our disposal, '`cp`'. If you, or your system administrator, has wisely set aliases for the `cp`, `mv` and `rm` commands so they are interactive for users (requiring you to respond with a "`y`" or "`n`" when removing or overwriting files), this example will work as presented. If not, substitute '`cp -i`' for the '`cp`' in the example presented to force the command into interactive mode.

## `yes` to the rescue

To illustrate this example, picture this... you've been working on copies of your web site's pages, and once everything checks out with all the changes you've made, you need to copy those files back to their permanent home. Using the interactive '`cp`' by itself, you would be faced with responding to each one of those "`overwrite `file' ?`" questions, one by one. You have a lot of files that you've worked on and don't want to spend all day pressing "`y`" to confirm each choice since you're copying interactively, as this can take quite some time. You also run the risk of skipping files if you're too fast with your response, type the wrong key, or even accidentally press `Enter` twice, as the command safely defaults to "`n`" and no action is taken with the file when that happens. The whole process, when working on a large group of files, can be more than slightly laborious in its own right. That's where the dandy utility, '`yes`' comes to our rescue.

Taking the scenario above, let's say those files you've been editing for your web site are in the directory "`working`" located in your home directory. You've also got the web site's original files in "`www`", also in your home directory. From your home directory you wish to copy the modified files to the web directory. To carry out this task, using '`yes`', you would issue the command:

```
$yes | cp working/* www/
```

Now, I'm sure that those of you that have been using GNU/Linux for a while will wonder "Why bother... why not just use '`/bin/cp working/* www/`' instead, since calling '`cp`' with the full path bypasses any aliases that may have been set in `/etc/bashrc` or `~/.bashrc` and does the

same thing?" Well, yes [pun intended], but I wouldn't have had such a good example, common on every system, to demonstrate with! As a matter of fact, since the above example continues to display its output on STDOUT (though not quite so orderly), some may prefer it for visual inspection. If they don't wish to see the output they could always append '`2> /dev/null`' to the end of the command, which would send any display to STDOUT to the bit bucket instead (though in this example the extra work wouldn't be worth it... just use `/bin/cp` instead).

## More than just a simple "y"

Now that you've seen the basic use of '`yes`', and knowing that GNU/Linux and *NIX utilities rarely stop short, I'm sure you won't be surprised to know that '`yes`' offers much more than outputting a simple "`y`". As a matter of fact, the command will take a string as an argument as well. Knowing this, the following command:

```
$yes The GNU/Linux utilities are wonderful, and powerful too!
```

will issue, you guessed it, the entire string following "`yes`" to STDOUT, over and over until the stream is broken. Remember though, when you pipe the output of '`yes`' to another command, the output is issued only as many times as needed to satisfy the requirements. In other words, if there were 50 files in your `~/working` directory that you were copying over to `~/www`, the output of '`yes`' would only be issued once for each file being copied. `Ctrl-C`, or `Ctrl-Break` is only needed when the command is invoked by itself, not piped. On a side note, should you ever need to generate a large nonsense file on the fly for testing purposes, the above example, with output redirected to append to a file and run for just a couple seconds will do it for you... *very* quickly. Have your fingers ready for the break when you press `Enter`, or you're likely to fill up your partition or quota in a flash!

## The ayes have it

As you can see, this handy little utility is very easy to use and adds a little more flexibility to your arsenal of utilities. The ability for '`yes`' to issue strings taken as an argument makes it perfect for those scripts or cron jobs that utilize interactive commands.

Although you might not want to use it with the interactive aliases for the `cp`, `rm` and `mv` commands since they can invoked directly from `/bin` as mentioned, I'm sure you'll run across that one interactive utility or filter '`yes`' would be perfect for, or find a use for it in your scripts or cron jobs sooner and later and be glad you have this simple utility at your disposal. Remember, there's more to '`yes`' than meets the "aye"! ;)