;login:
THE MAGAZINE OF USENIX & SAGE
February 2003 • volume 28 • number 1

inside:

SYSADMIN
Simmons: Symlinks and Hard Links Don't Belong in /etc

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

symlinks and hard links don't belong in /etc

by Steve Simmons

Steve Simmons has been an active member of the UNIX system admin community for two decades, including stints as president of SAGE and chairman of LISA.

scs@di.org

They don't belong in several other places as well, but let's for the moment just talk about /etc and look at some problems they introduce.

There are an awful lot of OSes out there that do their startups in System V style. To be more specific, they have a lot of /etc/rcX.d directories, where X is a run level like 0–6 or S. These directories have files like K20lpd, which starts up lpd.

So to modify run level 2, you cd to /etc/rc2.d and edit the files, right?

Wrong, as we all (should) know. In Solaris, all the various rc*.d/K20lpd files are hard links. Change one, you change them all. Well, depending on what editor or change technique you use. Recently in sage-members someone posted a problem with a Perl script that changed files in place and would end up replacing symlinks with new files while leaving the file the symlink referred to untouched. But is that bad? The answer is, it depends on what he wanted.

Take System V style rc files as an example. In Solaris, there are various /etc/rcX.d directories, where X is the various run levels available. Inside both /etc/rc0.d and /etc/rc2.d are files like K20lpd, which starts the line printer daemon.

In both run-level 0 and run-level 2, the line printer daemon does exactly the same thing. Sure enough, the two K20lpd files are one file with multiple hard links. So when you change it in place, you change the line printer characteristics in both run levels.

And sometimes you're changing unexpected things as well. For example, in Solaris 2.6, K20lpd has four hard links. Only two are in /etc/rc*.d/K20lpd. The other two are elsewhere. We'll come back to that in a minute.

Things can start biting you quite hard at that point. Most of us are smart enough to do backups of a system configuration file before changing it. The method you use to do that backup will affect the result of your changes. I like to do this:

```
# cd /etc/rc2.d
# mv file file-orig
# cp -p file-orig file
# vi file
```

This preserves everything about the original file, including inode modification dates. If you decide to undo it later by doing

```
# mv file-orig file
```

you've restored the system as nearly as possible to its original state.

But when symlinks and hard links are involved, things get dicey. If file is a symlink to something else, the file you're editing is the local copy only, not the other copy in rc0.d. Is that what you intended? Well, did you mean to change all lpd performance or just the rc2.d performance? Did you? Odds are good you didn't think about it. Until you've been bitten a few times, anyway.

An alternative (and potentially simpler) method of doing the same preservation is

```
# cd /etc/rc2.d
# cp -p file file-orig
```

```
# vi file
```

But when you restore it, you'd better remember to do

```
# cp -p file-orig file
```

rather than

```
# mv file-orig file
```

because if hard links are involved, you're going to get different results. So be sure and do it the same way consistently. And you should make sure all the other admins you work with do it the same way.

Different editors can introduce new issues. Emacs often makes automatic backups. Does Emacs preserve the inode numbers, mod dates, etc., when you revert the file? Do you *know*? What about vim? Pico? All versions of those editors? And what about your local Emacs customizations?

And what if you keep your rc files under RCS or CVS? Do they do in-place restores for hard links? Symlinks? And what about rdisting?

After a while, the experienced sysadmin knows to be careful about mucking about with the rcX.d contents. Unfortunately, that's not the only place we have to deal with this sort of thing. /etc/termcap is a symlink on many systems. So is /usr/man. You'd better just get in the habit of watching for symlinks and hard links any time you edit a configuration file, right? Unfortunately, yes. This little issue – the presence of symlinks and multiple hard links in configuration directories – leads to a morass of potential problems. Simply put:

Symlinks and multiple hard links don't belong in configuration files.

Period. They should be avoided whenever possible. And they can be, usually via a mechanism that makes the system *more* maintainable, not less.

Solaris 2.6, bless its pointy little head, almost gets it right. In 2.6, K20lpd has four hard links. They are:

```
/etc/init.d/lpd
/etc/rc0.d/K20lpd
/etc/rc2.d/K20lpd
/etc/rc2.d/S80lpd
```

What's needed is to get rid of the last three copies and instead replace them with something obvious, such as this shell script:

```
#!/bin/sh
#
# Strip the directory from the path and leading XNN from the name
# and run the core script in the base directory.
#
SCRIPT=`basename $0`
STRIP=`echo "$SCRIPT" | sed 's/^[A-Z][0-9][0-9]//'`
BASE=/etc/init.d
if [ "$STRIP" = "$SCRIPT" ] ; then
      echo "'$SCRIPT' is not a properly formed RC script name. Skipping." 1>&2
else if [ ! -x "$BASE/$STRIP" ] ; then
      echo "'$SCRIPT' base executable ('$BASE/$STRIP') not found. Skipping." 1>&2
else
      "$BASE/$STRIP" $@
fi ; fi
```

Symlinks and multiple hard links don't belong in configuration files.

symlinks and hard links in system areas are a recipe for problems with system administration.

When you go to edit /etc/rc0.d/S80lpd and find that script, you've just been alerted that you're looking at a shared configuration. You still face the decision of whether to modify the master ($BASE/$STRIP) or copy it to the local dir and modify just that one, but now you can't avoid making a conscious decision rather than letting the editor determine where the chips will fall. And no matter how you backed it up, cp or mv or CVS or RCS, the right thing will happen when you restore.

The current implementation of Solaris rc*.d files was a conscious decision on someone's part. The author was seduced by the attractive solution of hard links, and we sysadmins get to live with it. The proposed solution above gives almost all the benefits of the current system, but without the potential for easy error.

Unfortunately, others aren't as easy to fix. A quick glance at /etc in Solaris 2.6 shows almost 70 symbolic links. Almost all have the same bad reason for existence: backwards compatibility to old OS versions. Some are sheer laziness on the part of the vendor. For example, any link from /etc to ../sbin is present only so that old shell scripts didn't have to be updated. That's 53 links in /etc/ that can simply go away. Similarly, another batch are files that moved to /etc/inetd or /var/adm more years ago than I care to remember. On my Solaris 2.6 box, cleaning up just the obvious ones would get rid of all but three of the symlinks at the top of /etc.

Some of the ones left are things the vendor might have less control over. /etc/termcap is a good example. It is a symlink to /usr/share/lib/termcap. Its original move was done for space-saving reasons in /. By replacing it with a symbolic link, neither the OS vendor nor the third-party vendors had to make a change – even binaries continued working properly.

But how many vendors are still shipping those binaries to run on current systems? Zero, I'd wager. That's not to say the code has been fixed. With the backwards-compatible symlink in place, no one has any inducement to track down those legacy references to /etc/termcap and fix them. This would mean work even for system admins, whose users might have TERMCAP=/etc/termcap in various shell initialization files. But once those are fixed, those same admins will never have to go back and repair the broken links or mangled backup copies.

Frankly, symlinks and hard links in system areas are a recipe for problems with system administration. IMHO they are present only because somebody didn't bother to fix the *whole* system when making an improvement. We have been living with the unfinished 1% (and the problems they cause) ever since. It's time to fix it. We admins should fix our legacy user definitions (e.g., TERMCAP); that's easy enough to do.

The problem with symlinks and hard links is easy to state. We need to state it to the vendors and back it up with sample fixes such as provided above. Individual admins can move this problem along by complaining to the OS vendors. SAGE can help by stating the problem and carrying proposed solutions as the collective voice of the admins.

Fixing the third-party vendor issues are harder. But most OS vendors have existing processes by which they sunset certain OS features. We need to identify things such as /etc/termcap that need to be placed into that sunset process and push to get them there. Assuming it's not required by POSIX, there's no reason that /etc/termcap should have persisted this long. Simple recompiles should fix that particular problem; if we can get the link removals into the sunset processes, we'll eventually seen cleaner systems out there.

It can be done better, it should be done better. But it won't unless we complain about it.