by Peter Collinson, Hillside Systems

STEPHEN SCHILDBACH

Solaris 8

My personal "nonwork" project for the Easter weekend was to upgrade my Ultra/10 to run Solaris 8. In the United Kingdom, we theoretically get a four-day weekend over Easter; shops don't close like they used to over this period but most offices do shut down for the duration. Being a computerholic, I often use longish holidays to stop doing "real" work and use the time for changing or learning about my personal environment. The box containing Solaris 8 arrived in March and I felt it was time to upgrade.

I like to track Sun Microsystems Inc.'s releases reasonably closely, largely because it's a good way of picking up many bug fixes that don't make it into the extensive patch system. Sun's Web site is exhorting people to upgrade, citing 10 good reasons for the change. I think that if you look at the list from the point of view of an end user who is sitting at the workstation and doing the work, then Solaris 8 perhaps doesn't appear to deliver much in terms of revolutionary change. Lack of massive change is probably a good thing, because it means that things are not massively "broken" either.

The release comes with a bunch of what might be termed "systemy" things, which are aimed at making the systems administration somewhat more GUI-centered and also giving administrators the ability to run your machine remotely. I find the efficacy of these systems hard to judge because as a person who just runs a single machine, I don't really need them. So I've picked out some things below that seem to have an impact on me as a desktop user.

Something Old

One bug I am pleased to see fixed in Solaris 8 is the ability to change the key repeat speed in X. This stopped working some time ago. It was dysfunctional in Solaris 7 and maybe also in Solaris 6. I have always liked to be able to hold down arrow keys and have the cursor move quickly around the file I am looking at, or press and hold delete and have it zip back to remove the offending text. The X server supports two time values that control key repeat. The first (`ar1`) is the interval that must elapse before the key repeat kicks in and defaults to 500 msecs, the second (`ar2`) is the interval that elapses between each repeated character (default 50 msecs). These defaults seem much too sluggish for me and I like to reduce them to 250 and 25 msecs, respectively.

Sadly, these values can only be set by supplying a special argument pair on the `Xsun` command line when the X server is started. CDE has made this reasonably easy. Step one is to copy the file `/usr/dt/config/Xservers` to `/etc/dt/config/Xservers`. The rationale here is that the file in `/etc` will be retained when you upgrade the system, but an upgrade may possibly overwrite the default copy of the file in `/usr/dt/config`, losing local changes. Step two is to reboot the machine. I found that unless the system sees the file in `/etc/dt/config` when it starts,

putting the file there has no effect. It's just ignored. This was deeply confusing, and even the guy at Sun support was phased when I produced a `truss` sequence that proved my system was not even looking for the file. Once you have rebooted with the file in place, the start-up program finds it and begins to use it. After that, you just need to log out and in again to play with values that are passed to the X server when it is started.

To set up the repeat values, you edit the last line of the file that invokes the `Xsun` program, adding

```
-ar1 250 -ar2 25
```

then log out and in again. The X server, `Xsun`, should now be running with these values, and you should be able see that by using the `ps` command. You may need to run the Berkeley version (`/usr/ucb/ps`) and add the `w` option to get a wide listing of the running commands. Also, of course, you should discover that holding down any key causes repeats to happen considerably faster than before. Be careful about setting `ar1` to be too short. You'll find that your fingers rest on the keys when you are typing, and it's quite possible to set things up so that you cannot type a single character, all you get is repeats.

Something New

If you read last month's column (see "Performance," June 2000, Page 28, `http://sw.expert.com/C2/SE.C2.JUN.00.pdf`), you'll remember that Solaris has generally retained information about open files in the system virtual memory. Although this has meant faster access to frequently used files, it was implemented using the same memory pool that was used to store parts of running programs. So if you left the system doing something that was file-intensive, you would return and have to wait for your programs to be paged into memory before you could get on with the work.

In Solaris 8, Sun has made changes to the virtual memory and paging system to alleviate the problem. It has implemented what it calls the *Cyclical Page Cache*. Essentially, separate lists of free memory are maintained by the kernel; one for file system activity and one for the other objects that require memory management, like applications, uninitialized application data, the kernel and shared libraries. When there is a heavy file demand, the file system no longer throws applications out from the working memory. It just throws out the pages that relate to file system activity.

Sun says the new system is much cleaner and faster. I don't disbelieve Sun, I am sure it has the statistics to prove this claim. Solaris 8 certainly seems to have stopped that unpleasant wait that used to occur on some occasions while you sat waiting for the system to page process images into memory.

Solaris 8 has several "latest editions" of some tools. For example, it is supplied with Netscape Communicator 4.7, which at the time of writing is Netscape's most recent browser edition. Netscape 6 is in beta test. Interestingly, I notice that HotJava, Sun's Java brows-

er, has gone onto the "we intend this to die" list.

There is also a new debugging system, `mdb`, that is set to replace the old `adb`. I've not had time to play with it as yet.

Something Borrowed

In what is quite a departure for Sun, Solaris 8 is shipping with many publicly available utilities. In fact, I've found that my personal `/usr/local` tree has needed significant pruning to remove duplicate entries. Many programs that I use or have used for years are now bundled with the system and no longer need to be pulled from the Net and compiled. Perhaps the movement to do this started with Solaris 7, which shipped with Van Jacobsen's `traceroute`.

Some of these programs are convenience items, like the `less` command, which is a better `more` text paginator and named so you could say, "less is more than more, and more is less than less." I suspect that `less` dates back more than 10 years. Another program is the email pretty printer, `mp`, which I can date to 1995, if not before. This program (and its derivatives, like the file printer `filep`) was originally written by Steve Holden, a U.K. Sun employee. You'll find that the complete `gzip` and `gunzip` suite is also present on `/usr/bin`, reflecting, no doubt, the widespread use of these tools for file compression and expansion.

Perl Release 5.005_03 is installed in `/usr/bin/perl`. This file is actually a hard link to the binaries that are located in `/usr/perl5/bin`. You'll find the other Perl command-line utilities in this directory, too.

Solaris 8 ships with a CD called the "Software Companion." This unloads a bunch of GNU tools into `/opt/sfw`, including a complete `gcc`-based C and C++ development environment, GNU `emacs` as well as Ghostscript. All of which saves a lot of humping binaries or source over the Internet to beef up the Solaris development environment.

Finally, the Apache Web server is now standard on Solaris 8.

Something Blew

Well, with all this in mind, I carefully looked at the installation guide for the system. These books are somehow always confusing and hard to follow. I imagine they are difficult to write because they have to cope with many different environments and options.

I initially wanted to upgrade my system from Solaris 7 to Solaris 8, figuring that it would make life easier if I didn't have to mess with an entire new system setup. The documentation seemed to indicate that I would be able to upgrade my system. All I needed was a 256-MB swap partition that could be used to install a temporary system, which would in turn do the job of file installation. I figured that to give me the necessary space I could amalgamate two disk partitions on my second disk, the swap area and the small unused "spare" root next to it . The total space was something like 298 MB, which ought to have been enough. After some trepidation, and calling Sun support, I determined that it was OK to relabel live disks using

the /etc/format command, and I did so to generate what the manual said was appropriate space.

However, the upgrade would not happen. I booted from the CD as advertised, but the install code resolutely refused to give me the upgrade option. I began to wonder if the installation was broken and started to pull apart the installation process, which luckily is a shell script or two. It turned out that it needed 387 MB (or thereabouts) of space, I determined this by inserting lots of echo statements in the shell script to find out what it was doing. Well, I didn't have a single unused partition that could deliver this amount of space.

I decided to move things about on my disks to create the needed empty partition, but while doing this, I managed to mistype the following:

```
# newfs /dev/dsk/c0t1d0s0
```

For some reason that is still beyond me, I replaced the last zero above by a "2," and without thinking too hard, said "y" to the "do you really want to do this?" question. I watched the command execute and realized it was generating a file system for a partition that was much bigger than the one I'd been trying to make. Realization dawned. I'd managed to write a new file system occupying the whole space on the second disk on my system. I'd written garbage into the two partitions containing live data that lived on the second disk. The data on the disk was dead, deceased, it was no more. It was no longer sensible to consider upgrading and so with essentially a defunct Solaris 7 system, I went for a complete system install of Solaris 8.

I had been prudent. There were valid copies of the information on my system written to DAT using Solstice Backup. I'd also created dump (now called ufsdump) tapes of the system partitions before I started messing around. I wished I had created a full dump at this stage. Sun had told me to do so and I had ignored it. Read and obey the instructions.

Now it seems to me, after the event, that Sun's installation system hadn't helped me. First, it could have told me that it needed 387 MB to perform the upgrade. Second, it could have been a little clearer that it only really deals with single disk systems. It does say this, somewhere, but not in large friendly type so that you notice.

However, on balance, maybe the full install was the way to go. I now think my disks are laid out somewhat better. The default installation for Solaris 8 splits your single disk into three chunks. There's a swap partition of 512 MB, a root partition of 1.7 GB and the rest is assigned to /export/home. Unlike previous system installations, where the root partition is tiny, the root here is intended to hold everything except the user files.

I tend to always partition my two disks the same way, so each disk has same-sized sections, which is taking the easy way out in reality–no decisions about what might happen in the future are needed. I do prefer to have large partitions because when you have loads of small sections, it's hard not to waste space. Some parts of the file system will fit into the small partitions and some will not.

Anyway, I decided that the root partition the default installation had given me was quite full when I had finished unloading the CD. Once the system was live, I mounted /var on the equivalent 1.7-GB root partition on my second disk and moved /opt into /var/opt, creating a symbolic link from /opt. Today, I've got nearly 2 GB of files spread across these two partitions and I expect this occupancy to grow as I patch the system and my backup files end up in /var/sadm. So the default setting for the root partition is a little small. Incidentally, I also moved /export/home onto my second disk, with the thought that it's a good idea to have user files on one spindle and system files on the other.

Getting Things Going

Once I had booted the system, I disconnected the network connection because this machine is the main mail host on my network and I didn't want it to receive any mail until the system was properly configured. Getting the correct configuration files in place was not as hard as I had feared. Nearly all system tailoring is done in /etc, so the trick was to pull the old copy back from the dump tape into some other place on the disks and use the old values as a reference, while adding in the various bits of tailoring that are needed.

I think in the end that a complete install of the system was a good thing. It helps not to have debris from the old system to clear up, and also means that you tend to reassess any changes that you'd made to the old system before automatically applying them to the new one.

All my user files were stored using Solstice Backup, which needed to be completely reinstalled. Recovering the user files meant that I had to recover the indexes. The system seemed relatively happy that its control files were now in a different place. I followed the instructions on the useful disaster recovery PostScript file that is included on the CD and soon the index files were back in a new place on the system.

Actually, a weakness in Solstice Backup is that it's much too reliant on the current file system structure. Pulling back the odd file is not easy when the original file system structure has disappeared. If you have enough disk space, you can cope by pulling back a whole "save set," and then wandering around the retrieved file system tree to find the file you need. My advice, if you have the choice, is to use ufsdump and ufsrestore. I certainly wish I had taken the time to completely dump my system, rather than thinking that I would manage to leave certain parts of the file system intact during the installation process.

In a reasonable time, I had recovered the system to a point where I felt it was OK to plug the network back into the machine and start using it. I've had no problems running old Solaris 7 and 6 binaries on Solaris 8. My old /usr/local went up intact and running, however, as I said above, I've been pruning it since to remove duplicate copies of various programs that I've installed into it in the past few years.

One interesting problem has been my printer. I have a Hewlett-Packard Co. 4Si/MX printer plugged into my LAN. I bought this printer some time

ago and it has some features that I like to use. First, it can do double-sided (duplex) printing if requested; this saves trees and is a good thing. Second, I load good quality paper into its bottom cassette and less good quality paper into its top cassette. When I print a letter or something that needs to be on nice paper, I can select the paper source and print using the bottom cassette.

Also, I don't want the printer to print a banner page for every print job. Because there is only a small number of people using the printer, banner pages are a dramatic waste of paper. Configuring the printer as a sink for Berkeley format remote spooling (which the Jetdirect card will do) has never been very satisfactory because eliminating the banner page has to be done in the queuing program. I think on SunOS, I aliased the `lpr` command to automatically add the program option that suppressed the banner.

On previous Solaris systems, I've pulled code from HP's Web site that worked with the standard printing system to drive the printer over the network. I do seem to remember that I had problems getting this system to install properly on Solaris 6 and 7. However, it was worth struggling with at the time because the system gave me access to a set of options to the `lp` command that allowed me to use double-sided printing and select the paper source.

Well, it became evident that I was not going to get the system going on Solaris 8. I was not sure why, perhaps the Solaris spooling system has now diverged too much from the standard System V installation, or perhaps the binary programs supplied by HP would no longer run. However, it *is* advertised as a Solaris 6 solution and things have moved on a bit. Anyway, I was about to give up in some disgust at the general complexity of the printing system and its intractability, when I noticed Sun has a new program available as part of the printing system that's designed to drive printers over the network.

Of course, this filter is not documented with a manual page. I do wish Sun would document things. Last time I complained in these pages about Sun not generating manual pages for programs on the system, someone in Sun mailed me with an explanation about the nature of the particular program I had complained about. But he said that I really didn't need to know about this program. Well, I disagree. I'd like to know about everything on my system. In truth, there are a considerable number of programs lying around in public `bin` directories that are completely undocumented and really ought to be.

I suppose to be fair, the net printing program is somewhat hidden on `/usr/lib/lp/bin/netpr`, but it's useful to know about it, and especially I'd like to know how to drive it properly and what it can tell me about the printer. For example, does it monitor the connection and obtain error messages back from the printer?

The spooling system Solaris has inherited from System V seems incredibly complex. However, when a new printer is installed into the system, a standard shell script is placed in `/etc/lp/interfaces` that is intended to talk to the printer. The script is then renamed to be the name of the printer, but is a copy of `/etc/lp/model/standard` or `/etc/lp/model/netstandard`, depending on how you've configured things. The script handles banner generation, multiple copies and filtering of the information. Filtering is important. If you send plain text to a PostScript printer, it needs to change the text into something that the printer can understand.

I decided that I could strip down the standard net printing interface script to hand off most of its complexity to the shell script that HP had supplied to drive the printer in its own spooler environment. The HP shell script appears to be much hacked on and is intended to be operating system-independent.

The HP script does most of its output to its standard output channel, so all I had to do was attempt to replicate some of the settings that were presumed to be loaded by the HP spooler system when it invoked the script. I removed some of the calls that were used to set things up for printers that are directly attached to serial lines. I also removed a lot of HP-UX tailoring code, in an attempt to see the wood for the trees.

Once the shell script is stripped down, I can call it from a modified interface script, capture its output and pass it into the `netpr` program that sends it over the network to the printer. The interface script needs changing to remove all the processing that it can do, while retaining its control functions. Care is needed to ensure that it returns sensible values back to its calling program. I found that it could get stuck, being called again and again because the spooler considered that the printing had not happened.

Quite a lot of shell script hacking went on to make all this happen. In the end I was successful, and now have a spooling system that does the job for me. It is somewhat tailored to my printer, the HP system knows about the type of printer it is driving.

Finally

Well, I am now happily running Solaris 8 and beginning to use some of its new facilities. Sun has reacted to the open-source revolution by announcing that it will be making the source of the system available in third-quarter 2000. There are clearly defined licensing terms that stop you from passing code onto third parties (you can find out more info on `http://www.sun.com`). But the ability to look at the source and find out how things work will certainly help me use the system more effectively. Sun has already changed its licensing terms for the binary distribution to make it available at no cost, again under a set of terms that you need to read carefully. I am beginning to wonder whether I should look seriously at running the Intel version on some of my other systems, so I can have the same system running everywhere. We shall see.

Peter Collinson *runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever... He writes, teaches, consults and programs using Solaris running on an UltraSPARC/10. Email:* `pc@cpg.com`.