

# Inside Solaris™

Tips & Techniques for users of Sun Solaris

## Version-controlled Web pages with CVS

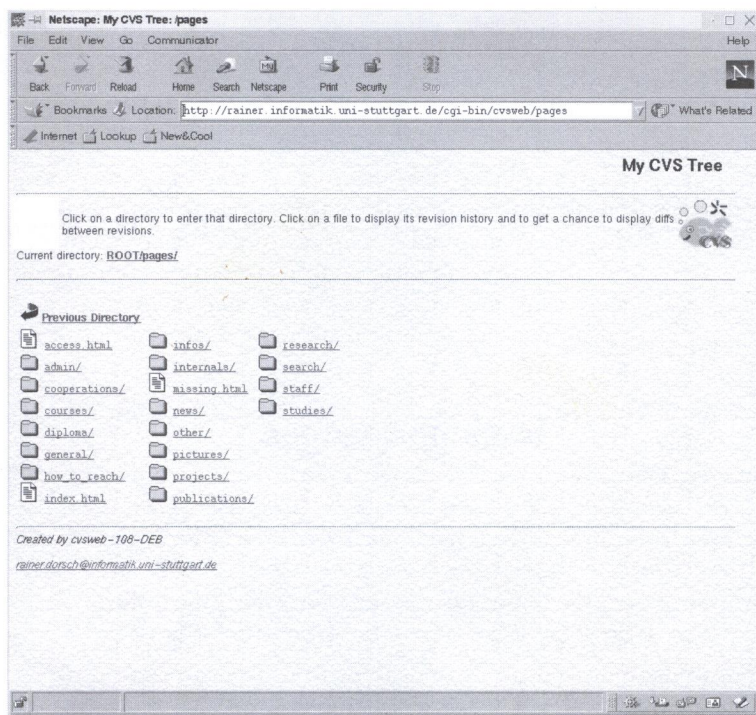
by Rainer Dorsch

A small group of editors of a division of a small company often manage the company's Web pages. One way to set up the division would be to name a master editor, who is the only editor with write access to the pages. The other editors would send him their suggestions or patches for the pages. The drawback to this is a considerable com-

munication overhead, especially if changes are small, such as fixing typos. Further, the editor requesting a change has to keep it in mind until the master editor completes the change.

Another extreme way to set this up would be to give write access to all editors. The drawbacks here are that pages can be overwritten if two editors edit local copies of a page at the same time. Further, it's difficult to find the editor who's responsible for a specific change in the pages. It isn't even guaranteed that an editor notices that there was a change at all, if it has been sufficiently small. These kinds of problems also occur in large software projects. Most projects use version control systems to handle these difficulties.

In this article, we'll describe how to use the Concurrent Versions System (CVS), shown in **Figure A**, to manage your Web pages. CVS is free software distributed under the terms of the GNU Public License (GPL). In our experience, once CVS was set up, it reduced the coordination overhead between editors



**Figure A:** The cvsweb interface shows us the directory structure of our Web pages.

### In this issue:

- 1  
Version-controlled Web pages with CVS
- 5  
Pining RMI servers
- 7  
A look at the Solaris 7 kernel
- 8  
**Quick Tip:** Adding a new device to your Sun
- 9  
Penetration testing and intrusion detection
- 12  
**Net.Update:** Upcoming Internet events
- 12  
**Solaris Q & A:**
  - Changing the login message
  - Apache detective work
  - Coosing your Java environment
  - A publishing system that runs under Solaris
  - Where's my new drive?
  - Adding swap space without a new partition



considerably. Additionally, it allowed each editor to be the master editor, but all changes were always trackable and documented.

The resulting overhead is the necessity of documenting changes in the Web pages (which could turn out to be an advantage in the long run). Overwriting changes made by another editor is impossible. Importing existing pages is no problem. In this article, we assume that you have a working Web server, like Apache, available.

## Editing the pages

Using the concept that each editor works on his own local copy of Web pages, this copy can be edited and browsed by the editor before it's installed on the Web server. CVS ensures that nobody overwrites changes made by somebody else, and documents which changes were done by which editor. For this reason, CVS has the master copy in the repository. Each editor has to supply a comment on why he performed the change.

Editing pages consists of three steps. We assume that the top-level directory of the local copy of the Web page is in the directory `~/www`:

1. Synchronize the local copy with the repository before editing pages. This is done by

```
cd ~/www
cvs update
```

This step is optional, but helps to reduce conflicts.

2. Edit the pages as usual with whatever method you want to use. We decided, for several reasons, to edit plain HTML at our site.
3. Commit your changes—that is, tell CVS that it should incorporate your changes in the repository. This is done by `cvs commit file`. If you don't give a file, the complete subdirectory tree is checked for changes and committed, if there are changes. Your favorite editor (specified by the `EDITOR` environment variable) pops up and asks you for a comment, which should be stored with the change. Note that it's not important to give the reason for the change. Just describe the change in words, like X replaced by Y. For small changes, like a typo fix, you can

give the comment, like the following, on the command line:

```
cvs commit -m "Typo fixed." file.html.
```

Usually committing your changes works. But from time to time, another editor commits a change in a page while you are editing it. In this case, CVS will reject your commit and ask you to synchronize your local copy first with the master copy (step one). When running CVS update, two things can happen:

- CVS merges your changes automatically with the changes the other editor committed. This might happen if the other editor changed a link in the page and you inserted some text in another location. We've experienced that automatic merging works for Web pages even better than it does for software.
- A conflict occurs. This might happen if you changed a link to a given URL and the other editor changed the same link to another URL. In this case, note in the file what conflict occurred. In this file, both alternatives of the conflicting section are shown in the following form:

```
<<<<<< file.html
Your alternative
=====
Alternative suggested by the other
editor
>>>>>> 1.6
```

If the reason for the conflicting change isn't obvious, you should first check the documentation, in which the editor of the conflicting change should have stated why he performed the change. If this doesn't solve your problem, contact the editor who committed the conflicting change.

## Tracking changes

Each night, we perform an automatic update of our Web pages from the CVS repository with a cron job. You can also force an immediate update after each commit by an editor using the `commitinfo` configuration file of CVS. The line in our crontab file is:

```
5 3 * * * cd /www; cvs update -d -P
```



All the editors get an email of the output of the update process, documenting which pages were changed. When an editor notices a change in a part of the Web pages he's interested in, he'd use the cvsweb interface to check who did which change for what reason (although the command line interface could also be used). cvsweb displays the directory structure and the files of the Web pages, as shown in **Figure A**. For each file, the history and the differences between files can also be listed, as shown in **Figure B**.

## Setup

To set up CVS for managing Web pages efficiently, begin by downloading the CVS binary from <http://sunfreeware.com>. Use `pkgadd -d <packagename>` to install the binary on your system. One exception in which you'll want to compile your own version of CVS is when you want to use cvs wrappers (see sections "Automatic tasks" and "Further information").

The first thing you should do is set up a repository. You have to select a directory (which should have an excellent backup) for the repository. Let's assume you choose `/usr/local/cvsroot`. Then you must initialize the repository using the following command:

```
cvs -d /usr/local/cvsroot init
```

Next, you'll want to import your existing Web pages (if you want to start from scratch, just omit this step). Let's assume you have a copy of the pages in `~/www`. The commands

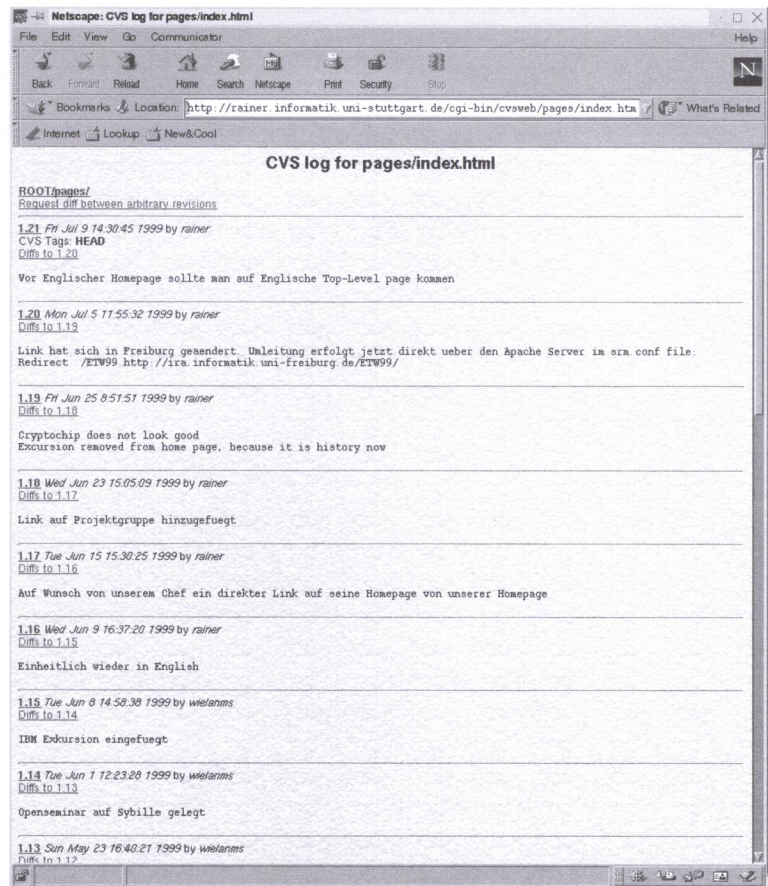
```
$ cd ~/www
$ cvs import -m "Imported sources" www
editor start
```

import your pages. Your pages will appear in the repository in `/usr/local/cvsroot/www`. The string `editor` is a vendor tag, and `start` is a release tag. They aren't important for our use of CVS. Now the setup of the repository should be complete.

Now, each editor should check his local copy of the pages. Make sure that you don't have a `www` subdirectory in the local directory

```
cvs -d /usr/local/cvsroot checkout www
```

This checks the local copy of the pages. You can now start editing the pages as we described.



**Figure B:** We can also display the history of each file's updates.

## Automatic tasks

Currently, we run two tasks automatically to enhance our installation of CVS. First, we update the last change line when the pages are committed automatically. We insert the date, a shortcut for the editor, and his email address. You do this by inserting the line

```
*.html -t
' /usr/local/share/wwwAdmin/scripts/
updateEditor %s %s'
```

into the `cvs wrappers` file. Wrappers allow you to set a hook that transforms files on their way in and out of CVS. Our line in `cvs wrappers` defines that the `updateEditor` script will run on each file whose name matches the filter `*.html`. **Listing A** on page 4 shows the `updateEditor` script.

The `webEditors` file contains lines with tab-separated fields containing initials, username, and email address. An example would be:



```
rd rainer rainer.dorsch@informatik.
uni-stuttgart.de
```

We also automatically generate a what's new page, which contains links to the new content on our site. Since we don't want each fixed typo to appear on the what's new page, we explicitly trigger the insertion into the news page. Our Perl script, `cvs2www.pl`, analyzes the comments that are given when a file is committed and generates a list entry for each comment with a leading plus sign (+) in the following form:

```
date: comment (editor)
```

Comment is the comment given in the commit process (without the leading plus sign character), and it's linked to the committed page. Editor is the initials of the editor, and is linked to his email address. The script is derived from Karl Fogel's `cvs2cl.pl`, which generates a GNU style ChangeLog file from the CVS log entries.

If you want to use the `cvs2www.pl` script to generate your news page, change the page header enclosed between the following lines:

```
my $WebPage_Header = <<'END_OF_HEADER';
[... ]
END_OF_HEADER
```

Change the first part of the footer enclosed between the following lines:

```
my $WebPage_Footer = <<'END_OF_FOOTER';
[... ]
END_OF_FOOTER
```

Finally, change the second part of the footer written by the following line:

```
printf NEWS_OUT "Last change...
```


The second part is separated because the current date is inserted automatically.

## Further information

If you have to set up a Web server first, an Apache binary for Solaris can be downloaded from <http://sunfreeware.com/>. Documentation about the Apache server can be found at the homepage of Apache [www.apache.org](http://www.apache.org).

We described only a small fraction of CVS functionality. You can find further information about CVS in Per Cederqvist's manual ([www.loria.fr/cgi-bin/molli/wilma.cgi/doc.847210383.html](http://www.loria.fr/cgi-bin/molli/wilma.cgi/doc.847210383.html)), the info pages coming with CVS, or Cyclic's Web page [www.cyclic.com](http://www.cyclic.com). There's also a mailing list info-cvs (see [www.cyclic.com/cvs/lists.html](http://www.cyclic.com/cvs/lists.html)) on which all kinds of CVS-specific questions are discussed. It's a low traffic list with 5-10 postings per day.

A CVS binary for Solaris 2.6 can be downloaded from <http://sunfreeware.com/>. Unfortunately, at the time of this writing, the current version of CVS doesn't support `-t/-f` cvs wrappers due to a bug in the code. CodeFab's version of CVS does support it. You can download the source code and Solaris binaries from [www.codefab.com/cvs.html](http://www.codefab.com/cvs.html).

We didn't discuss graphical user interfaces for CVS. There are various options for different platforms, including all kinds of UNIX, Microsoft's operating systems, and MacOS. See [www.cyclic.com/cyclic-pages/software.html](http://www.cyclic.com/cyclic-pages/software.html). You can download Karl Fogel's ([kfogel@red-bean.com](mailto:kfogel@red-bean.com)) `cvs2cl.pl` from [www.red-bean.com/~kfogel/cvs2cl.shtml](http://www.red-bean.com/~kfogel/cvs2cl.shtml) or our modified version to generate our news page from [www.ra.informatik.uni-stuttgart.de/~rainer/Download/cvs2www.pl](http://www.ra.informatik.uni-stuttgart.de/~rainer/Download/cvs2www.pl). The GNU Public License (GPL) is available at [www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html). 

### Listing A: Script to transform our checked-in and checked-out HTML files

```
#!/bin/bash

export LC_ALL=en_EN.ISO-8859-1
webEditors=/usr/local/share/wwwAdmin/data/webEditors

EDITOR='grep $USER $webEditors!cut -f1'
MAIL='grep $USER $webEditors!cut -f3'
SEARCH='Last change:[^\(]*(
```



# Pinging RMI servers

by Atiq Hashmi

Java provides a simple distributed object application facility called Remote Method Invocation (RMI). Using RMI, applications can distribute Java objects to other machines to run within servers where the client can invoke methods of those objects just like a local call.

However, RMI is a simple, remote method invocation facility and doesn't provide mechanisms to manage or monitor the servers. One ability that isn't directly available is being able to see that the servers that have exported their services are still listening for requests. In other words, there's no utility to ping an RMI server. In this article, we discuss a Java programming scheme to provide such a ping service.

## RMI overview

An RMI-based system consists of a client and a server. The server provides services by defining objects, usually referred to as remote objects, and methods associated with them. It then exports its object to a registry process called *rmiregistry*, which serves as a name service for the servers. The *rmiregistry* runs on a well-known port, which is 1099 by default. This exporting process is called *binding* to the registry. Each server registers or binds itself with a unique name (sometimes called a *binding entry*) in the registry.

The client uses two steps to use the remote server. First it gets a reference to a remote object by looking up a server by its binding entry. Once it has that reference, it can invoke any methods on it and get the service.

However, finding a binding entry doesn't necessarily mean that a server is listening on it. This happens when, for example, a server dies without unbinding itself. The client application only finds this after invoking a remote method and failing, as a result.

## Java interfaces

A client and server agree on the services by defining a Java interface file. In Java, an interface defines a protocol of behavior that is essentially a set of methods that the server must implement. A Java class is then defined that contains the actual implementation code for

those methods. Interfaces can be extended by subinterfaces, just like classes.

The following is an example of a Java interface definition:

```
public interface testinterface
↳ extends Remote
{
    public String testHello() throws
        RemoteException;
}
```

Here, the interface defines that the server that will implement this interface will provide a remote method called `testHello()` that takes no arguments and returns a string as a result.

## Using the ping interface: Server side

As discussed, in order to know if a particular server is actually running, you need to actually invoke one of the remote methods. The scheme we present here is to define a separate generic ping interface that's extended by all other application interfaces. The advantage of this is that the ping facility becomes a uniform and standard facility across the product development environment. Besides, any other rmi server management services can be added as part of standard remote services (for example, a method to collect server usage statistics, etc.).

We also provide a sample implementation code for the ping method that can be used by interface implementation classes. This sample code provides useful information, such as the user that owns the server, the time the server started, the host the server is running on, etc. **Listing A** on page 6 shows the generic ping interface.

The applications can provide any implementation they like, but one way to make the ping system work is that the application interfaces extends the `PingIfc` interface shown in **Listing A**. Since the `PingIfc` extends the `Remote` interface, the application interfaces need not extend it again.



In the class that implements the application interface add an instance variable like this:

```
private String startTime_ = new
Date().toString();
```

### Listing A: Our generic RMI ping interface

```
// PingIfc.java

import java.lang.String;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface PingIfc extends Remote{
    public String ping() throws RemoteException;
}
```

### Listing B: Setting up our Ping

```
//classes needed for the Ping remote method
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.lang.SecurityException;
import java.util.Date;
/**
 * This function provides ping feature
 * to the clients.
 */
public String ping()
    throws RemoteException{

    InetAddress ia = null;
    String s = null;
    try{
        ia = InetAddress.getLocalHost();
    }catch(UnknownHostException e){
        e.getMessage();
        System.exit(1);
    }catch(SecurityException e){
        e.getMessage();
        System.exit(1);
    }

    s =
    System.getProperty("BINDING_NAME") +
    ": " + System.getProperty("user.name") +
    ": " + ia.getHostName() +
    ": " + System.getProperty("PORT") +
    ": " + startTime_;
    return s;
}
```

Also, add a ping() method to the class. **Listing B** shows a sample implementation of this method that creates a string containing some ping-related information. The top part shows the classes that need to be imported in the file. When a client invokes this method, the method prepares this string and returns it to the client. The example assumes that the values for binding name and the registry port are available in the system properties.

You could add other pieces of information as well in this string. If this information doesn't change during the life of the server, the construction of this string could be implemented so that it's constructed just once.

## Client side

Now that we have a ping service implemented on the server side, you can write a tool to invoke the ping() method to check whether or not the server is listening. However, before the ping() method can contact the server for information, there could be other problems—for example, the rmi registry may not be running, or there may be no binding entry in the registry. The tool needs to detect these before invoking the ping call. Java specifies different exceptions that the rmi runtime throws when an error occurs. We'll briefly discuss how to use the ping meaningfully by first checking these errors.

## Using the ping meaningfully

The first API to use is Naming.list(), which lists all the servers that the rmi registry knows about. If the rmi registry isn't running or not running on the expected port, a ConnectException is thrown. If the list of binding entries is successfully found, the next API used is Naming.lookup(), passing each of the list results obtained one by one in the argument.


If there's no binding entry in the rmi registry, a NotBoundException is thrown. However, if the lookup() successfully returns a reference of the remote object, the ping() method is invoked on this object. If the server is listening, a ping response with the server information is returned, as discussed; otherwise a RemoteException is thrown.

Note that the listed exceptions are only meant for explanation. The Java API documentation lists all the exceptions that may occur for these calls, and Java requires that most of those must be caught by an application.



## Conclusion

We presented a way to ping RMI application servers and get useful information about the

servers. As Java evolves, we may see additional Java APIs and facilities providing more capabilities for the RMI application environments. 

# A look at the Solaris 7 kernel

by Edgar Danielyan

At the heart of the Solaris operating environment is the kernel, as in all UNIX systems. However, unlike other UNIX systems, the SunOS kernel isn't a single, monolithic executable file, but a structured collection of files. In this article, we'll take a look at the Solaris kernel façade and kernel directory structure, as well as its contents.

## The structure and the contents

As mentioned earlier, the Solaris (or SunOS) kernel isn't a single file, but a collection of files in /kernel and /platform, and their subdirectories. As you may guess from their names, /kernel contains common kernel files, and /platform contains platform-dependent parts. Lets start with /kernel:

```
# ls /kernel
drv      exec    fs      genunix
└─misc   sched  strmod  sys
```

All of these are directories, except genunix, which is the kernel itself. It's an executable in ELF MSB format, statically linked. If you do an ls in the drv directory, you'll see a number of files with and without the .conf extension; these are the drivers themselves. Note that there are also virtual drivers—that is, drivers for non-physical devices, such as ip and arp drivers.

The accompanying \*.conf files are their configuration files (as you've already guessed). Generally, you don't need to modify anything in them; if you do modify, make sure you understand what you want and what you do, because wrong configuration files will cause problems. The drivers are ELF MSB executables, and the .conf files are text files.

The next directory is exec, which contains only three files:

```
aoutexec elfexec  intpexec
```

These are program loaders, which execute programs stored in three corresponding formats, AOUT, ELF, and INTP. There are no configuration files here and nothing that needs to be taken care of.

A perusal of the next directory, fs, will reveal that it contains file system drivers—one file for each file system type:

```
Autofs  fifofs  lofs    procfs  specs   ufs
Cachefs          hsfs   nfssockfs  tmpfs
```

No configuration files here, either.

Let's move on to the misc directory, which contains miscellaneous modules, such as md5 and des, which implement MD5 and DES encryption algorithms, accordingly. The sched directory contains the system scheduler itself. The strmod directory accommodates System V STREAMS modules—there are no configuration files here.

The last directory in /kernel, sys, hosts various system-loadable modules. For example, kaio is the kernel asynchronous I/O module; nfs implements the NFS; semsys and shmsys implement semaphores and shared memory subsystems, respectively.

Now, on to /platform. See Listing A on page 8. All SUNW.\*s are symbolic links to sun4m, because the models listed are of a common architecture, and that architecture is called sun4m. It contains two files, kadb and ufsboot, and a directory, reasonably called kernel. The kadb is the kernel debugger, and the ufsboot is the UFS file system boot loader.



The kernel in turn contains other subdirectories and one file:


```
cpu   drv   misc   strmod  unix
```

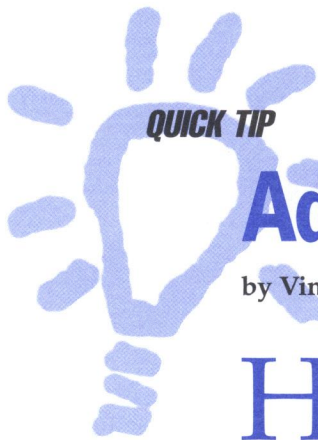
**Listing A:** *The platform directory listing*

```
# ls /platform
SUNW,S240
SUNW,SPARCstation-10,SX  SUNW,SPARCstation-LX+
SUNW,SPARCclassic       SUNW,SPARCstation-20   SUNW,SPARCsystem-600
SUNW,SPARCclassic-X    SUNW,SPARCstation-4    SUNW,Sun_4_600
SUNW,SPARCEngine-EC-3  SUNW,SPARCstation-5
SUNW,SPARCstation-10  SUNW,SPARCstation-LX
Sun4m
```

The only file here, *unix*, is the sun4m architecture-dependent part of the kernel. Directories *drv*, *misc*, and *strmod* are also sun4m-dependent counterparts of the same directories in */kernel*. *cpu* contains CPU codes for various processors, the default being the Sun micro-SPARC processor (in case of sun4m architecture). There are also files for Ross- and TI-made processors.

There's one higher-level kernel configuration file, the */etc/system*, which isn't described here, but we'll cover it in a separate article.

This is just a small part of the roadmap to Solaris. As you can see, the Solaris architecture isn't one big monolith, but actually a series of pieces that are designed to work together. 



**QUICK TIP**

# Adding a new device to your Sun

by Vinay Gupta

**Test**  
your software  
**Knowledge**

**H**ow many times do you need to reboot a Sun machine (running Solaris) after adding a tape drive, CD-ROM drive, or hard drive? The answer is *zero*! This may surprise quite a few of you, but it's true, thanks to Sun's *drvconfig* located in */usr/sbin/drvconfig*.

*drvconfig* can create the full */devices* directory hierarchy that describes the hardware of that machine. You can also use *drvconfig* for a selective update to the */device* directory hierarchy. Once you have all device entries, then getting a link to */dev* is very easy. You can create all the links by hand, or you can use the *devlinks*, *disks*, and *tapes* commands to create them.

Here's a simple example showing how you can add a tape device to a running Sun machine, without any down time:

1. Find out the used scsi IDs. `ls /dev/dsk/*s2` will give you all the scsi IDs used for hard drives and CD-ROM drives, and `ls -l`

`/dev/dsk/?` will give you all the scsi IDs used by existing tapes.

2. Set the tape scsi ID to an unused one on the system.
3. Connect the tape drive at the end of the scsi chain and Power on the tape drive.
4. Rebuild */devices* for the new scsi tape device by using `drvconfig -i st`.
5. Make */dev* links by using *tapes*.
6. Check for the correct entry with `ls -l /dev/rmt/0`, if this is your first tape drive on the machine.
7. Insert tape and try `mt -f /dev/rmt/0 stat` for a further check.

This same set of commands can be used to recover a crashed OS, where part or full of */devices* and/or */dev* directory are missing. In this case, boot the machine via the Solaris CD-ROM or the network. Or you can take out



the disk and mount it to some other machine. Now use the `-r` option (available in all commands) to rebuild the `/devices` and `/dev` directory tree.

If you have mounted the crashed OS disk on `/tmp/mnt` then you have to:

1. `drvconfig -r /tmp/mnt`
2. `devlinks -r /tmp/mnt`
3. `disks -r /tmp/mnt`
4. `tapes -r /tmp/mnt`

That's all there is to it! 

# Penetration testing and intrusion detection

by Paul A. Watters

Security is a constant concern for networked information systems. The requirement of ensuring data integrity must always be weighed against the usability of systems and interfaces, in the face of constant threat of attack from the outside world.

In this article, we'll take a fresh look at an old adage in the Solaris world—the best way to determine if your system is secure is whether or not you can break into it. This kind of penetration testing is becoming increasingly common for Solaris systems, as administrators attempt to stay one step ahead of the wily cracker.

However, an often overlooked aspect of computer security is effective intrusion detection—a prerequisite to even determining whether systems have been compromised. In this article, we'll review the state-of-the-art in intrusion detection, focussing on some simple, manual procedures, which act as a first line of defense against intrusion. These measures can be used to detect deliberate penetration tests, as well as attacks from the outside.

## Why do they hack?

Although obtaining data by breaking into or cracking networked computer systems is a serious crime in many countries, many people are willing to risk hefty fines and imprisonment. Why? Forget the popular media image of a precocious teenager trying to break into NASA. Computer crime by experienced and

organized syndicates is a profitable and endemic problem that will become worse as the number of networked computers exponentially increases.

Breaking into computer systems isn't a game—it can cost an enterprise thousands or even millions of dollars to repair damaged systems, and more seriously, cause financial loss through leaking of trade secrets and/or direct interference in trusted financial transactions.

Since many of the world's key computer systems run Solaris, it has been a popular target, with postings to USENET publicizing bugs in third-party software and freeware causing panic and disruption to operations. We have all read about buffer over-runs in C programs, exploiting race conditions in multi-threaded applications, as well as browser bugs which reveal sensitive local information. The two questions which naturally arise are, "How can we protect our systems from intrusion?" and "How do we know when our systems have already been compromised?"

## Security packages

There are many commercially available security packages that mine key system data logs for irregularities of various kinds in TCP connection requests, which might indicate suspicious or fraudulent access. However, there's a likelihood that an experienced cracker will be aware of these programs and their scanning techniques. Instead of initiating a port scan in sequential order every five seconds, a cracker



might decide to generate a scanning pattern that scans unusual ports randomly or chaotically, interspersed with accesses to normal ports (like port 80 for HTTP requests).

Intrusion detection systems can provide valuable data about persistent attempts to log on, or deny access to your computer, by attempting to consume all available bandwidth on your network connection by issuing fake broadcast requests (smurfing). However, the best solution is to get a feel for who logs on to your system when, and what kind of activity levels are representative of legitimate processing during different times of the day.

## Detection

One of the best tools for monitoring logins and access to systems is the TCP wrapper package, which pre-dates Solaris 2.x. This package was designed to provide a substitute for each network daemon, which called the real daemon on request after logging the request to a system log file. This greatly increased the awareness of administrators about use of their system. In addition, alerts could be produced on attempts to access any one of the network daemons: for example, attempts to rsh or rlogin could be monitored.

Of course, the best solution to reduce the potential for misuse of these network services is to switch them off explicitly in `/etc/services`, and/or filter packets at the firewall level destined for those services. This solution provides both internal and external protection of data through port connections. There may also be situations where particular ports must be available locally for use (for example, for database listeners) but must be denied to external users. A firewall is definitely necessary in this situation, although many routers these days have a basic packet filtering capability.

Another important tool in detecting intrusions on your system is tripwire, which monitors local filesystems for changes in size and other characteristics that might indicate a Trojan horse or other attack. A signature of each filesystem is taken, and verification against future states takes place at regular intervals. Unexpected and unauthorized changes can be cross-referenced with access data provided by tcp wrappers, to track down the source and target of illegal access to data.

Access to networked systems varies in many ways, and there are many legitimate, freely

available services for which monitoring may not be necessary. A good example is an anonymous FTP server—or is it? Many implementations of FTP require a version of the password file to be present in the `etc` directory of the anonymous FTP area. Some administrators failed to read the fine manual for these systems, and unwittingly made the whole password file available by anonymous FTP. Obtaining a UNIX password file is like gaining the keys to the kingdom. Although the password fields are encrypted, password-guessing programs like Crack assume that many users base their passwords on variations of dictionary words, proper names, capital cities, and a number of other easily guessable sources.

## Reducing the risk of intrusion by gaining access to the password file

There are two ways to reduce the risk of intrusion by gaining access to the password file. First, a `passwd` replacement like `npasswd` should be installed, as it prevents users from changing their passwords to those which can be guessed by Crack using default dictionaries. The drawback is that users of the system will refuse to change their passwords because the acceptable alternatives are too difficult to remember, or worse still, they might write their password on a notepad next to their terminal or PC.

The second strategy involves enabling password shadowing, so that the password file doesn't actually contain a world-readable copy of all the encrypted passwords. Passwords are kept in a separate shadow file that's only accessible by the superuser.

## Penetration

In an article entitled "Improving the Security of Your Site by Breaking Into It," found at [www.nease.net/~ping/admin-guide-to-cracking.html/](http://www.nease.net/~ping/admin-guide-to-cracking.html/), Farmer and Venema suggest that the best form of defense is attack—an attack on your own system. Although this might seem like an unusual suggestion at first, the basis is that if you can easily break into your own system, then an intruder will also have little difficulty.

Alternatively, you can pay one of the many security consultants who specialize in this kind of defense to determine the vulnerabilities and weaknesses of your systems.



Whichever path you choose, you should always be aware of some of the basic strategies by which people obtain unauthorized access to systems.

One of the oldest tricks in the book is to try out so-called default system and application passwords to gain access to the system via a particular port or protocol, which might reveal further information about a target system. Some non-Solaris operating systems ship with the default superuser password combination, system/manager, which is also typical of some popular RDBMS systems.

Although gaining read-only access to a database through an external listener and a default username and password might not seem as serious as someone gaining full superuser privileges, commercially sensitive data might be easily revealed in this way. In addition, there's the constant threat that an intruder may subtly change some data in your tables, which might leave your company liable for damages arising from neglect, especially if a third-party client is involved. Deciding on sensible username/password combinations for both system and RDBMS accounts is a prerequisite for any serious level of security.

Another useful test is accessing your system with a low-level user account, perhaps as a guest user, and investigating what system features you can gain access to (especially write access). As mentioned earlier, some systems have world-readable password files, as well as lists of supported services, personal details about other users, and technical data on the size and capacity of a target machine. All of this information, once compiled, is fairly useful to an intruder.

Users on machines networked to your own might like to participate in an NFS and SMB mounting exercise, where they attempt to mount all available shares and volumes from your computer and see what information they can reveal. Was it really such a great idea to share "/", revealing your system password files to everyone in the company? Often, when the external threats of attack are minimized through the use of firewalls and other security layers, the issue of internal organizational trust becomes the most significant barrier to security.

Adventurous users may wish to switch their network card into promiscuous mode,


directly reading the bytes transmitted through the network. Some clever filtering software will allow you to capture the passwords transmitted in clear text over the network. It's better for us to be aware of these vulnerabilities, rather than a cracker who has gained control of the router between our mail server and workstation, reading off our POP password to read mail unaware that we have just surrendered control of the system. A better solution is for all data exchanges through the network (including password authentication) to be encrypted, using a secure-socket layer for Web-based services, and secure shell and copy for telnet-like services.

## SATAN

One of the most useful tools for testing the security of Solaris systems is SATAN, the Security Analysis Tool for Auditing Networks (and more recently, SAINT). Although criticized heavily when released for openly exposing the defects of many networked computer systems, SATAN managed to bring out into the open the whole issue of just how vulnerable our systems are to attack.

The software works by using several programs to systematically detect (and exploit) weaknesses in a target system (in this case, our own). Although SATAN itself has some security flaws, which have been highlighted, it's very useful for determining the nature of specific vulnerabilities in a single and/or a network of systems, which can provide the basis for an action plan.

## Further reading

The number of papers, software packages, and exploit scripts grows exponentially each month, and so it's difficult to recommend the best tools or sources of data regarding intrusion detection and penetration testing. One of the most up-to-date sources of information about weekly trends in cracking is the National Infrastructure Protection Center, [www.nipc.gov](http://www.nipc.gov), which posts a fortnightly summary of the most common attack types, viruses, and newly published exploits, ranked by potential seriousness of outcomes for servers (for example, a low ranking for denial of service, but a high ranking for root access). The USENET forum comp.security.unix also contains current information, as well as lively debates, about security issues and industry trends. 



# Upcoming Internet events

by Edgar Danielyan

## The Internet Society

Draft Charter of the Non-Commercial Domain Names Holders Constituency (NCDNHC) of the Domain Name Supporting Organization (DNSO) is published at [www.ncdnhc.isoc.org](http://www.ncdnhc.isoc.org). Comments and discussions are encouraged and welcome. This draft was written by Kathryn Kleiman, Don Heath, David Maher, and Randy Bush with input from many people.

## Conferences INET'2000

The date is set for INET'00, the premier international Internet conference. The Internet Global Summit, 10<sup>th</sup> Annual Internet Society Conference will be hosted by the Pacifico Yokohama Conference Center in Yokohama, Japan, on July 18-21, 2000. Mark your calendar now!

## NDSS

The Network and Distributed Systems Security Symposium (NDSS) will be held at the Catamaran Resort Hotel, San Diego, CA, on February 2-4, 2000. Among the speakers is Professor Eugene Spafford of the Purdue University, ACM Fellow, IEEE Senior Mem-


ber, widely known expert on information security.

## Reseaux IP Europeens

RIPE NCC published its 1998 Annual and Financial Reports, available online at [www.ripe.net/annual-report/98ar.html](http://www.ripe.net/annual-report/98ar.html).

Reseaux IP Europeens, American Registry for Internet Numbers, and the Asia Pacific Network Information Center published a common policy draft on the Address Supporting Organization (ASO) of the ICANN, which will be submitted to the ICANN. Public comments and feedback are encouraged.

## IANA

A document called "Internet Domain Name System Structure and Delegation" has been published jointly by the IANA and ICANN for public comment. The document summarizes current practices of the IANA in administering RFC 1591, which deals with the delegation and administration of country code top-level domains (ccTLDs), such as .US, .UK, and .FR. The full text of the document may be found at [www.iana.org/tld-deleg-prac.html](http://www.iana.org/tld-deleg-prac.html). Email comments to [comments@icann.org](mailto:comments@icann.org). 

## SOLARIS Q & A

### Changing the login message

*Changes to the law in my country require me to explicitly warn users logging on to my system that breaking into my computer system is illegal. What's the best way to do this?*

There are two possibilities: /etc/issue is a text file that's printed before the familiar

login prompt, and so can deter potential crackers from attempting to log in. Alternatively, /etc/motd is displayed after a login, and can be used, for example, to display internal usage policies (such as, developers must *not* attempt to access filesystems to which they haven't been explicitly given access).



## Apache detective work

I have an Apache Web server running with a lot of virtual hosts and redirects, and I find it hard to keep track of what my server is doing! Is there an easy way that I can get a listing of these kinds of details?

The configuration for an Apache Web server is traditionally kept in three separate files: httpd.conf (server configuration), srm.conf (resource configuration), and access.conf (authorization and security). The trend these days is to combine configuration, resource, and security information into a single httpd.conf file.

It's quite easy to use a Bourne shell or Perl script to search for the information you need and dynamically display the results to a text file or, more imaginatively, to a Web page. This makes viewing your current configuration

very easy. An example Perl script to do this is shown in **Listing A**.

After defining some basic information that's used to define virtual hosts, such as the IP address of the host (\$ip), the definitions for a configuration file (\$httpfile), and resource file (\$srmfile) are given. If these files exist and are readable, some HTML tags are generated to ensure a properly formed document. Next, the script searches through \$http file for occurrences of <VirtualHost \$ip>, in this case <VirtualHost 58.12.23.34>, and extracts the virtual hostname, terminating the line with a break.

A similar process occurs for \$srmfile, to find any redirect definitions. This script is entirely general, and other Apache parameters (like AddIconByType definitions) could also be extracted and displayed in this way.

### Listing A: Example Perl script

```
# ListApache.pl on ftp.zdjournal.com/sun

#!/usr/bin/perl
print "Content-type: text/html\n\n";
$ip="58.12.23.34";
$httpfile="/usr/local/apache-1.3.6/conf/httpd.conf";
open (HTTP,$httpfile) || die
↳"System error: Unable to process your request";
$srmfile="/usr/local/apache-1.3.6/conf/httpd.conf";
open (SRM,$srmfile) || die
↳"System error: Unable to process your request";
print "<HTML>\n";
print "<HEAD>\n";
print "<title> Virtual Hosting & Redirection
↳Report</title>\n";
print "</HEAD>\n";
print "<BODY BGCOLOR=#FFFFFF>\n";
print ("<h1>Virtual Hosting: $ip</h1>\n");
while (<HTTP>)
{
    if (/<VirtualHost $ip>/)
    {
        $found=1;
    }
    else
    {
        if ($found==1)
        {
            s/ServerName//g;
            print "$_."<br>";
        }
        $found=0;
    }
}
close(HTTP);
print ("<p><h1>Redirections $ip:</h1>\n");
while (<SRM>)
{
    if (/Redirect/)
    {
        s/Redirect//g;
        s/http/\n<br>\t=> http/g;
        print "$_."<br>";
    }
}
close(SRM);
print "</BODY>\n";
print "</HTML>\n";
```



## Choosing your Java environment

*I'm new to Java development, and I'm using Solaris 7 as my Java development platform. Currently, I'm using JDK 1.1.7, but I want to be able to test the new features of Java 2 (for example, swing) by installing the new JDK 1.2. Can I have both installed on my system at the same time?*

### **Listing B:** Changing our path to use JDK 1.2

```
JAVA_HOME=/usr/java1.1; export JAVA_HOME

PATH=$PATH:$JAVA_HOME; export PATH

Classes are located by setting the CLASSPATH in Bourne shell:

JAVA_LIB=$JAVA_HOME/lib/classes.zip; export JAVA_LIB

CLASSPATH=$CLASSPATH:$JAVA_LIB; export CLASSPATH
```

*How do Java programs know which package to use, and where to find classes?*

It's certainly possible (and desirable) to be able to continue development with JDK 1.1.7 while investigating the new features of Java 2. If you've installed the JDK 1.1.7 software package from Sun using `pkgadd`, it should have been installed in `/usr/java1.1` by default.

If you download the package for JDK 1.2 and install using `pkgadd`, Java 2 will be conveniently installed in `/usr/java1.2`. To force your Java programs to find a specific Java compiler, simply set your path to find it. In Bourne shell, assuming that Java isn't already in your path, this can be produced with a command, such as the one in **Listing B**. If you ever need to use JDK 1.2, simply substitute `/usr/java1.1` with `/usr/java1.2`.

## A publishing system that runs under Solaris

*I work for a small- to medium-sized enterprise (SME), and I'm required to implement an organization-wide publishing system (kind of like FrontPage for multiple users in a single work unit). Since my other server-side applications run under Solaris, I would like to install an application with a strong knowledge management focus, and which supports revision control. Can you suggest anything with a reasonable price tag?*

The first question organizations usually ask after discovering the possibilities of the Web is, "How can I get all my employees working cooperatively in this medium?" NT-based tools like Microsoft FrontPage ([www.microsoft.com](http://www.microsoft.com)) are fantastic for single users, but offer little in the way of document protection or user integration for distributed publishing.

Other products for NT, like DocuShare by Xerox ([www.xerox.com](http://www.xerox.com)), have powerful indexing and search capabilities for all different types of documents, but DocuShare is difficult to install and is written in the Python interpreted language (not exactly an industry standard).

One alternative approach is to take advantage of the emergence of Java as a write-once, run-anywhere network programming language, and develop a system that interacts with heavyweight commercial databases to store Web data. One such product is Intranet Solution's Intra.Doc! ([www.intranetsol.com](http://www.intranetsol.com)), which features revision control and automated Web publishing. Unfortunately, it comes with a fairly hefty price tag: \$100,000 for unlimited users. Although it runs on both Solaris and NT, it's also quite limited in terms of support Web servers and databases.

The best solution for SMEs and individual workgroups comes from Whitewolf ([whitewolfsoftware.com](http://whitewolfsoftware.com)), whose website-MAX product is priced at under \$4,000 for a 4-user license, including support. This product will work with any database that supports JDBC, and runs on any operating system that supports Java servlet technology. websiteMAX is written completely in Java, and supports template-based HTML development via an online, browser-based interface.



## Where's my new drive?

I've just installed a new hard drive, but my SPARCstation doesn't recognize that it's there! Isn't Solaris' plug-and-play like other operating systems? Do I need device drivers?

Your SPARCstation no doubt uses a SCSI bus, and you won't need special device drivers to install a hard drive. You must prompt the operating system, however, to reconfigure its internal listing of devices by issuing the command

```
touch /reconfigure
```

prior to rebooting, or by using

```
boot -r
```

to boot the kernel. If all goes well, you'll see messages confirming that the devices directory is being reconfigured. The next step, assuming that you know which partitions you wish to create for your drive, is to run the format program, and use the *partition* option to size new partitions. Using the

## About our contributors

**Edgar Danielyan** spent some time studying US and UK law and is working as a network administrator and manager of a top level domain of Armenia. He has worked for the United Nations, the ministry of defense, a national telco, a bank, and has been a partner in a law firm. He speaks four languages, likes good tea, and is a member of ACM, IEEE CS, SENIX, CIPS, ISOC, IPG, and many other lesser known organizations. He can be reached at [edd@computer.org](mailto:edd@computer.org).

**Rainer Dorsch** received his master in Physics in 1996 at the University of Ulm (Germany) and works now at the University of Stuttgart (Germany) on his Ph.D. in Computer Science. He started with UNIX in 1992, has worked with Linux at home since 1994, and is heavily involved in Solaris administration since 1996. When he's not working, he likes hiking, cycling, or swimming in one of the beautiful lakes in the Alps, or working in his orchard. He welcomes your comments and criticisms. You may reach him via email at [rainer.dorsch@informatik.uni-stuttgart.de](mailto:rainer.dorsch@informatik.uni-stuttgart.de).

**Vinay Gupta** works as a UNIX/NT system administrator with a consulting company. He has been in this field for almost eight years. His resume can be found at [www.members.tripod.com/vinay\\_k\\_gupta/resume.html](http://www.members.tripod.com/vinay_k_gupta/resume.html).

**Atiq Hashmi** is a software engineer at Telcordia Technologies (formerly Bellcore). His interests are in system and network administration, applications and tools, as well as Internet-related technologies. He can be reached at [hash100@mail.eclipse.net](mailto:hash100@mail.eclipse.net).

**Paul A. Watters** is a research officer in the Department of Computing, Macquarie University, Australia. He can be reached at [pwatters@mpce.mq.edu.au](mailto:pwatters@mpce.mq.edu.au).

## Coming up...

- Cross-platform development
- Using IMAP on Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by  
ZD Journals 500 Canal View Boulevard, Rochester, NY 14624.

### Customer Relations

US toll free .....(800) 223-8720  
Outside of the US .....(716) 240-7301  
Customer Relations fax .....(716) 214-2386

For subscriptions, fulfillment questions, and requests for group subscriptions, address your letters to

ZD Journals Customer Relations  
500 Canal View Boulevard  
Rochester, NY 14623

Or contact Customer Relations via Internet email at [zdjcr@zd.com](mailto:zdjcr@zd.com).

### Editorial

Editor .....Garrett Suhm

Assistant Editor .....Jill Suhm  
Managing Editor .....Joe Froehlich  
Copy Editors.....Rachel Krayer  
Christy Flanders  
Taryn Chase

Contributing Editors .....Edgar Danielyan  
Rainer Dorsch  
Vinay Gupta  
Atiq Hashmi  
Paul A. Watters  
Print Designer .....Rachel King  
Melissa Ribauda

You may address tips, special requests, and other correspondence to

The Editor, *Inside Solaris*  
500 Canal View Boulevard  
Rochester, NY 14623

Editorial Department fax .....(716) 214-2387

Or contact us via Internet email at [sun@zdjournals.com](mailto:sun@zdjournals.com).

Sorry, but due to the volume of mail we receive, we can't always promise a reply, although we do read every letter.

### ZD Journals

General Manager..... Kelly Baptiste  
Director of Customer Support ..... Doug Neff  
Director of Operations..... Kress Riley  
Manager of Design and Production ..... Charles V. Buechel  
Manager of Product Marketing ..... Mike Mayfield  
Product Manager ..... Brian Cardona

### Postmaster

Periodicals postage paid in Rochester, NY and additional mailing offices.

Postmaster: Send address changes to

*Inside Solaris*  
P.O. Box 92880  
Rochester, NY 14692

### Copyright

Copyright © 1999, ZD Inc. ZD Journals and the ZD Journals logo are trademarks of ZD Inc. *Inside Solaris* is an independently produced publication of ZD Journals. All rights reserved. Reproduction in whole or in part in any form or medium without express written permission of ZD Inc. is prohibited. ZD Journals reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for personal and commercial use. For reprint information, please contact Copyright Clearing Center, (978) 750-8400.

Inside Solaris is a trademark of ZD Inc. Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, SunInstall, OpenBoot, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. Other brand and product names are trademarks or registered trademarks of their respective companies.

Printed in the USA.

### Price

Domestic .....\$99/yr (\$9.00 each)  
Outside US .....\$119/yr (\$11.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703.

### Back Issues

To order a back issue from the last six months, call Customer Relations at (800) 223-8720. Back issues cost \$9.00 each, \$11.00 outside the US. You can pay with MasterCard, VISA, Discover, or American Express. Collections of back issues are available on CD-ROM as well. Please call for more information.

## Are you moving?

If you've moved recently or you're planning to move, you can guarantee uninterrupted service on your subscription by calling us at (800) 223-8720 and giving us your new address. Or you can fax us your label with the appropriate changes at (716) 214-2386. Our Customer Relations department is also available via email at [zdjcr@zd.com](mailto:zdjcr@zd.com).



Sun Technical Support  
(800) 786-7638

## PERIODICALS MAIL



\*\*\*\*\*3-DIGIT 480

C: 7661905 00002096 04/00

20  
34

Please include account number from label with any correspondence.

command `newfs` will create a new UFS file system on each of your new partitions, which can be mounted using the `mount` command. Don't forget to enter the appropriate settings

into `/etc/vfstab` if you want your new partitions to be available after the next reboot (but you don't need to reboot your SPARCstation for your drives to be available!).

## Adding swap space without a new partition

*I'm running low on physical RAM, but I need to run some applications temporarily that require lots of memory. Can I add swap space on Solaris without creating a new swap partition?*

Creating a separate partition for swap space is probably the optimal solution for Solaris. But it's possible to create temporary swap by creating an empty file of a particular size, and then adding its capacity to the existing swap pool.


To examine the current state of swap space allocation, use the `swap -s` command. Output from the command will look something like this:

**Listing C:** Output from `swap -l`

swapfile	dev	swaplo	blocks	free
/dev/dsk/c0t0d0s5	32,5	16	2097344	539472
/dev/dsk/c0t3d0s1	32,25	16	1052144	125312

```
total: 1298112k bytes allocated + 272440k
➔reserved = 1570552k used, 369824k
➔available
```

On our machine, there's clearly a lot of swap memory allocated, but quite a bit is already allocated to applications. In order to view the existing configuration of swap space, use the `swap -l` command. This produces output like the table in **Listing C**.

This shows two partitions (`c0t0d0s5` and `c0t3d0s1`) that are set aside specifically for swap. If we wanted to add around 100 MB, we'd first create an empty file by issuing the `mkfile 100M /tempswap` command. This would create the file `/tempswap`. To add the file as swap space, use the `swap -a /tempswap` command. Of course, when you've finished using the temporary swap space, it's easily removed by issuing the `swap -d /tempswap` command. 

ZD Journals  
On the  
**MOVE?**

## Take us with you!

If you're moving on to a new job or a new location, be sure to take us along! Email your new mailing address along with your customer number to [ZDJCR@zd.com](mailto:ZDJCR@zd.com) or fax us at (716) 214-2386.