

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Transparent email

by Richard Auletta

Solaris offers a variety of user interfaces for accessing email. Unfortunately, when Solaris connects to the Internet via an Internet service provider (ISP) that assigns a dynamic IP address, many of these applications no longer function. Instead, only email applications that support the POP3 protocol can be used to pop mail off the ISP's mail server. This article offers a solution to allow your favorite email readers to work with dynamic, on-demand TCP/IP links, just like they do with a permanent IP address and host-name. Our solution is outlined in **Figure A**.

Overview

When accessing the Internet over a PPP or SLIP connection through an Internet service provider, the ISP will typically assign a dynamic IP address and name to your computer. The IP address and name can change each time the PPP connection is made, making it impossible to have email delivered directly to your system, since it doesn't have a permanent name as far as the Internet is concerned. Instead, email is delivered to your ISP's mail server and moved to your local computer via the POP3 post office protocol. Outgoing mail is sent to the ISP via SMTP, the simple mail transport protocol.

POP3

A very popular POP3 mail application for Solaris is Netscape Communicator. It supports popping

mail down via POP3 and sending mail out via SMTP. Both full-time connected and PPP users use this excellent mail interface. But you might find that this approach has some shortcomings. Because your mail is popped down to a local mailbox, your `.forward` file, vacation setup, filter, and procmail automatic email processing won't work. Of course, `CDE dmail`, `Openwindows mail tool`, and a host of other mail interfaces can't be used either, expecting the delivery and transmission of email to be performed by `sendmail`, a mail transport agent.

Fetchmail

An alternative to using a mail user application (MUA) like Netscape Communicator is to use `fetchmail` to pop your email. `Fetchmail` can use POP3, IMAP, and a host of other

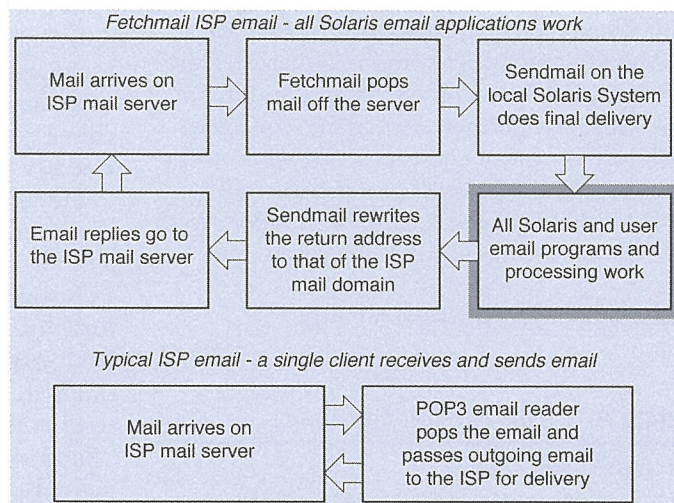


Figure A: This illustrates the difference between using our fetchmail solution vs. a typical ISP mail server.

In this issue:

- 1** Transparent email
- 4** Quick tip: Defining a multi-line prompt
- 5** Connecting Solaris to the Internet via dial-up PPP
- 8** All about SunPC cards
- 12** Watching your logs
- 15** Solaris Q&A:
 - Is the process still alive?
 - What is a memory leak?
- 16** Tool of the Month: Network top

protocols to fetch your mail from a mail host. The difference is that `fetchmail` delivers the incoming mail via SMTP by talking to your `sendmail` daemon or other mail transport agent (MTA), instead of just dropping it into a mailbox file. In other words, `fetchmail` acts like a `sendmail` daemon on a remote machine and makes the mail appear to be delivered normally through your `sendmail`—allowing all your MUA and MTA features to work.

Of course, the real upshot of using `fetchmail` is that it allows the use of any Solaris email application to access email. Popular command line email programs such as `elm`, `pine`, or `mutt` will work, as will CDE's `dtmail` or Openwindow's `mailtool`. Even Solaris `dcm` and `cm` calendar managers can send mail to remote machines once you've properly configured `sendmail`.

`Fetchmail` has some very convenient features, the best of which is that it will run as a daemon in the background, popping your email at regular intervals from the ISP's mail host. Manually running `fetchmail` while the daemon is active wakes up the daemon to check the server for mail.

`Fetchmail` has a large number of options that can appear intimidating at first glance, but it's typically easy to configure. The `.fetchmailrc` shown in **Listing A** will be sufficient for most ISPs. Simply change `your_isp_pop3_server` to your ISP's mail pop server. For US WEST, in Denver Colorado, it would be `pop.dnvr.uswest.net`. And, of course, change `your_password` to your actual password. `Fetchmail` won't run unless the permissions on your `.fetchmailrc` are 0600.

You can obtain `fetchmail` and read about its amazing list of features and capabilities from its official home page, <http://sagan.earthspace.net/~esr/fetchmail>. To compile `fetchmail`, start by unzipping the archive with `gzip` and untar-ing with `tar`, and then change into the `fetchmail-4.6.0` directory and type `./Configure`, and then `make`, and then `make install`. To install as root you'll likely need to add `make` to root's path with the following Bourne shell command:

```
PATH=$PATH:/usr/ccs/bin;export PATH
```

Listing A: Sample `fetchmail` `.fetchmailrc`

```
set daemon 300
poll your_isp_pop3_server protocol POP3:
password your_password;
```

When building `fetchmail`, you'll need the GNU version of `flex` available as a Solaris package or source from <http://sunfreeware.com> as `flex-2.5.4a-local.gz`. You can also pick up a compiled version of the GNU `gcc` C compiler at <http://sunfreeware.com>.

You can also pick-up a pre-built `pkgadd` version of `fetchmail` 3.9.5 from <http://sunfreeware.com> for Solaris 2.5. We experienced no problems with either a `gcc` compiled 4.6.0 version of `fetchmail` or the pre-built 3.9.5 `fetchmail` version on Solaris 2.6 when connecting to two ISPs.

Sendmail

While `fetchmail` will pop your email from the ISP mail host, outgoing email must be delivered via `sendmail`. Having configured `fetchmail`, you must now configure `sendmail` to make email completely transparent to your email applications.

Two problems must be resolved when delivering email via `sendmail` without a permanent IP address and name. First is that `sendmail` needs to rewrite your return address as the address of your email account that your ISP supplies. The other issue is that your machine's hostname as defined in `/etc/hosts` isn't registered as either a valid hostname or domain by the network information center (NIC) and won't resolve during domain name lookup. In other words, your hostname will be bogus as far as the Internet is concerned, and many sites won't accept email from an unresolvable machine name or with an unresolvable email address. Hence, your `sendmail` will need to masquerade as your ISP both in terms of the return address and email headers.

You'll have to upgrade to `sendmail` 8.9.1 or Solaris 2.7 in order to have `sendmail` masquerade as your ISP. You can get `sendmail` 8.9.1 from www.sendmail.org or from <http://sunfreeware.com> as a pre-built `pkgadd` binary. Upgrading to `sendmail` 8.9.1 won't only ease the configuration of the `sendmail.cf` file needed by `sendmail` to masquerade as your ISP, but will also give you a modern, secure `sendmail` implementation.

Installing `sendmail` 8.9.1 is relatively straightforward. First, download the gzipped tar file from www.sendmail.org. The next steps are to unzip it, untar it, and change into the directory `sendmail-8.9.1/src`. Then type `./Build`, and `./Build install` as root. The full documentation in postscript format can be found in the file `sendmail-8.9.1/doc/op/op.ps` and can be viewed with `pageview`.

An alternative to compiling `sendmail` is to grab a pre-built `pkgadd` binary from <http://sunfreeware.com>. Either way, you'll need to install both the new Berkeley DB package and the m4 UNIX macro processor. Both can be found at <http://sunfreeware.com> as `db-2.4.14-local.gz` and `m4-1.4-local.gz`, respectively.

Once you have `sendmail 8.9.1`, you'll need to create a `sendmail.cf` file to implement both host and envelope masquerading. Go to the `cf` directory of the `sendmail-8.9.1` distribution, and then into the `cf` directory within that directory. If you did a package install, the `cf` directory will be in `/opt/local/cf`. Copy the file `generic-solaris2.6.mc` into `your_host.mc` and edit the file to look like **Listing B**, changing `your_esp_domain` to your ISP's domain. For US WEST, it would be `uswest.net`. Check the README in the `cf` directory for details about building a custom `sendmail.cf` file.

Lines 2 and 3 will configure `sendmail` for a generic Solaris system. Line 4 will masquerade the email headers and line 5 the envelope. Lines 6 and 7 enable local deliver for local address and SMTP for remote address. Line 8 allows you to pick the domain name seen in the headers to make sure everyone realizes they can't return mail to your computer's `/etc/hosts` defined hostname. If you want to have your email relayed through your ISP, then add the following line, as shown in **Listing C**.

```
define('SMART_HOST', your_esp_smtp_server)dnl
```

For US West in Denver, Colorado, `your_esp_smtp_server` would be set to `pop.dnvr.uswest.net`.

There are advantages and disadvantages to relaying your mail through a relay host. The advantage is that the email will be delivered to your ISP immediately for relaying. If you don't relay and there are delivery problems, the mail will be queued for later delivery on your machine, and if the PPP connection drops, the email won't go out until the connection comes back up. On the other hand, having the local `sendmail` talk directly to the destination machine gives you more control and monitoring capabilities. The `mailq` command will allow you to monitor the status of outgoing mail, and your `/var/log/syslog` will show if the mail has been delivered to the final destination. To build the `sendmail.cf` after editing the `.mc` file, run the command

```
m4 ../m4/cf.m4 your_host.mc > sendmail.cf
```

in the `cf/cf` directory.

Copy this `sendmail.cf` to `/etc`, and the binary for `sendmail` to `/usr/lib` (if you haven't done so already), and restart `sendmail` from the `/etc/init.d` directory with the command `./sendmail stop`, then `./sendmail start`. You might want to edit the `/etc/init.d/sendmail` script to have `sendmail` run the queue more often than every hour. To have `sendmail` run the queue every 10 minutes, change the line `/usr/lib/sendmail -bd -q1h` to `/usr/lib/sendmail -bd -q10m`.

Make sure to add a fully qualified hostname to your `/etc/hosts`, or use one in the following statement in your `.mc` file.

```
define('confDOMAIN_NAME', 'not.a.real.machine')dnl
```

If you don't, `sendmail` will suspend for 60 seconds in a vain attempt to obtain one. A fully qualified hostname (FQDN) is simply a hostname with a domain attached (for example, `name "dot" domain.`) `Sendmail` doesn't care or know if the domain is real, but will try to obtain

Listing B: Sendmail m4 .mc configuration file

```
1: VERSIONID('@(#)solaris2.6.mc 8.9 (Berkeley)
  5/19/98')
2: OSTYPE(solaris2)dnl
3: DOMAIN(generic)dnl
4: MASQUERADE_AS(your_esp_domain)dnl
5: FEATURE(masquerade_envelope)dnl
6: MAILER(local)dnl
7: MAILER(smtp)dnl
8: define('confDOMAIN_NAME', 'not.a.real.machine')dnl
```

Listing C: Relay sendmail m4 .mc configuration file

```
VERSIONID('@(#)solaris2.6.mc 8.9 (Berkeley)
  5/19/98')
OSTYPE(solaris2)dnl
DOMAIN(generic)dnl
MASQUERADE_AS(your_esp_domain)dnl
FEATURE(masquerade_envelope)dnl
define('SMART_HOST', your_esp_smtp_server)dnl
MAILER(local)dnl
MAILER(smtp)dnl
define('confDOMAIN_NAME', 'not.a.real.machine')dnl
```

Web Resources

Fetchmail

<http://sagan.earthspace.net/~esr/fetchmail>

Sendmail

www.sendmail.org

Tin

www.tin.org

Freeware for Solaris

<http://sunfreeware.com>

a qualified hostname and will issue warnings when the hostname isn't fully qualified. An email address that isn't fully qualified can't be used as a return address.

If you're running Solaris 7, you only need to edit `etc/mail/sendmail.cf` as shown in **Listing D** after copying `etc/mail/main.cf` into `etc/mail/sendmail.cf`. Restart `sendmail` after making the edits. You might be tempted to set the `$j` macro at line 67 to your ISP's domain, but don't. If you do, `sendmail` will attempt to deliver all mail addressed to any user at your ISP's domain on the local machine.

You can check your installation by sending yourself some email at your ISP address using `mailx -v your_mail_address`. Executing `mailx` with the `-v` option will display the SMTP exchange between your `sendmail` and the ISP's `sendmail` daemon. Of course, then you'll want to fetch

Listing D: *Changes to sendmail.cf for Solaris 7*

```
Line 65: # my officil domain name
Line 67: Djnot.a.registered.domain

Line 72: # "Smart" relay host (may be null)
Line 72: D$your_isp_stmp_server


Line 112: # who I masquerade as (null for
          No masquerading) (see also &=M)
Line 113: DMyour_isp_domain
```

the mail and check that the return address is correct.

USENET News

Accessing USENET news is another issue that has to be resolved by the home user. Again, Netscape Communicator will allow you to read news via the network news transport protocol (NNTP). If you prefer a command line interface, the threaded USENET news reader `tin` is available at www.tin.org or as a binary from <http://sun-freeware.com>. The latest version works well over slow dial-up connections. Run `tin` with the option `-rn` to read news via NNTP and to only load newsgroups subscribed to in your `.newsrc`. Set the shell variable `NNTPSERVER` to your ISP's news server. For US West in Denver, Colorado, the news server is `news.uswest.net`.

Conclusion

You can make your home installation of Solaris more familiar and much more convenient by using `fetchmail` to automatically pop email from your ISP's mail server. `Fetchmail` will automatically fetch and deliver your email through `sendmail`, allowing all your favorite Solaris email applications to work with an ISP and a dynamic IP address. 

Defining a multi-line prompt

by Alvin J. Alexander

If you use a standard Korn shell `$PWD>` prompt when working at the command line, you might be running out of space to type, because the names of your directory paths take up most of the line. If most Solaris pathnames weren't long enough already, Internet and intranet servers make them even longer by pre-pending something like `/usr/local/lib/apache/htdocs` to the name of every Web directory you work in.

I solved this problem by switching to a multi-line command prompt. With my multi-line prompt, the current working directory is displayed on one line, while I work on a much larger command-line below the printed path. This way I can see the entire path, while still being able to have a long command-line to type on.

For example, when I'm working in the Apache `htdocs` directory, my prompt looks like this:

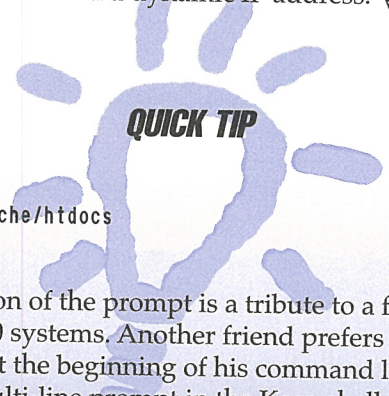
```
/usr/local/lib/apache/htdocs
==> _
```

(The `==>` portion of the prompt is a tribute to a friend who loves AS/400 systems. Another friend prefers the happy :) symbol at the beginning of his command line.)

Setting up a multi-line prompt in the Korn shell is easy. Just enclose the `PS1` prompt definition in single quotes, and hit the [Enter] key wherever you want a carriage-return to appear. Here's how I configure the prompt shown in my `.profile`:

```
PS1='
$PWD
==> '
```

The single-quote characters are the perfect fit for multi-line prompts, because they preserve the carriage-return characters within the `PS1` prompt variable.



Connecting Solaris to the Internet via dial-up PPP

by Peter Marelax

Are you tearing your hair out trying to connect your Solaris system to the Internet? Have you spent countless hours trying to get the Solaris-bundled PPP working? Don't give up now, there's a powerful yet free alternative called the PPP daemon.

The *PPP daemon* was developed by Paul Mackerras to connect many flavors of UNIX to the Internet using the Point-to-Point protocol. As such, it has been widely adopted as the de-facto PPP Link Manager in free UNIX operating systems, such as FreeBSD and Linux.

Requirements

The PPP daemon is a very versatile piece of software. Under most circumstances, it can be customized to meet the requirements of your ISP. Unfortunately, the way in which users connect to the Internet can vary with each ISP. Therefore, it would make this article very confusing and complex if we attempted to cover all scenarios. So, we'll focus on a common access method, supported by the majority of today's providers. This method possesses some requirements of your ISP's dial-up service. They include support for PAP authentication and Auto PPP detection.

You can safely assume you're in this category if you've previously connected to your ISP using Windows 95/98/NT, without utilizing the Dial-up Scripting or Bring Up Terminal Window options in Dial-up Networking. If you're still unsure, it's best to ask your ISP. If you don't fall into this category, further documentation is available once the software is installed. With this information, you can tailor the configuration to the specifications of your ISP. See the section titled "Further information," once you've followed the installation instructions.

Obtaining the software

The most difficult hurdle you have to overcome is obtaining the software, because it's primarily available via the Internet. For the sake of convenience, we've made Solaris 2.5.1, 2.6, and 7 Sparc and x86 packages available

on the Web at www.phaseone.com.au/Solaris/Packages/.

We'll endeavor to keep these up to date with the latest version of the software. If you're feeling adventurous, you can download and compile the latest source code yourself. It's available at the FTP address in the download icon above.

Installing the software

The first step is to install the packaged software. By default, the package is installed under `/opt/FreePPP`. To install it, run this command as the root user.

```
# pkgadd -d FreePPP-Solaris.x86.2.6.pkg
```

Once installed, you need to configure it for your system.

Configuring the software

All operations described throughout this article must be performed as the root user because the ownership of most files are set to the root user during installation.

Edit the file `/etc/ppp/peers/isp` using your favorite text editor. This file contains various options. We've made an effort to describe each option within the contents of the file. *Read this file carefully.*

Let's make the required modifications. Add your ISP's dial-in phone number by substituting the word *PHONENUMBER* with the actual number. Following on, we need to tell the PPP daemon to authenticate to the ISP using your account information. Find the option "user USERNAME" and replace USERNAME with your ISP account's login name.

Next, we need to inform PPP as to which serial or COM port your modem is connected to. Uncomment the line that reflects the following line, if your modem is connected to serial A/COM 1:

```
#cua/a
```

Uncomment the line that reflects the following line, if your modem is connected to serial B/COM 2:




```
#cua/b
```

Further down in the file you'll see references to modem speeds (for example, 115200, 57600, etc.). Uncomment only one of these lines to indicate the maximum DTE speed your modem is capable of. Typically, a 56K modem can peak at 115200, and 33.6K modems at 57600. That concludes the changes needed to this file, so save your work.

The last file we need to modify is `/etc/ppp/pap-secrets`. This file contains your login name and password. PPP will use this information to automatically authenticate with the ISP using the PAP protocol. That means you don't need to enter your account details every time you dial-up. PAP stands for *Password Authentication Protocol* and is supported by the majority of ISP's today. Edit the file and substitute username with your ISP's login name.

Finally, substitute password with the plain text password of your ISP account. Save your work and check the permissions on this file. Ensure it's readable by the root user only, as it contains your password!

The configuration of PPP is complete. Now we need to make some changes to Solaris, so it can communicate with your ISP and essentially the Internet.

Configuring Solaris

You're halfway to hooking into the Internet, but first we need to configure Solaris to use DNS as one of its naming services. Before we do this, you must obtain the IP address of your ISP's DNS server. Most ISPs inform you of your system's required settings when you sign up for an account, so search through the paperwork for details on DNS. Once you find it, modify, or create the file `/etc/resolv.conf` and add a line like so:

```
nameserver IPADDRESS
```

Substitute IPADDRESS with your ISP's DNS server IP address. If your ISP has multiple DNS servers, add additional lines. This will give your Net connection some level of redundancy if one of the ISP's DNS servers happens to fail.

Solaris now knows what DNS server to talk to, but we need to turn it on. Modify `/etc/nsswitch.conf` and locate the line that starts with `hosts:`. On a stand-alone Solaris system with no naming services enabled, this line

would look like `hosts: files`. Or, if you were running NIS+, it may look like:

```
hosts: nisplus [NOTFOUND=return] files
```

Whatever the case may be, we have to add the word `dns`. Essentially, you want to end up with something like:

```
hosts: files dns
```

This is for a system with no naming service. If you *are* using a naming service, then the syntax would be similar to:

```
hosts: nisplus files dns
```

This is for a system with NIS+ turned on. Adding `dns` tells Solaris to use DNS when the system attempts to resolve hostname's to IP addresses and vice versa. Now, save your work.

Finally, we need to enable a syslog facility. This will allow you to diagnose the state of PPP as you attempt a connection. Add a line like so in `/etc/syslog.conf`:

```
local2.debug<tab><tab><tab>/var/log/ppp.log
```

Don't type the letters `<tab>`. You need to hit the [Tab] key or this won't work due to a formatting error. Now, create the `ppp.log` file like so:

```
# touch /var/log/ppp.log
```

Next, re-initialize the syslog service by sending a HUP signal to the process ID of the syslog daemon. This command will do it for you:

```
# kill "HUP `cat /etc/syslog.pid`"
```

Logging is now enabled.

Testing your modem

The configuration of the software and operating system is now complete. But before jumping the gun, it's always a good idea to ensure that your system can happily talk to your modem. We're going to use the `cu` command to test the modem. For this example, I'll assume the modem is connected to serial B/COM 2 and has a maximum DTE speed of 115200. Start `cu`:

```
# cu "l /dev/cua/b "s 115200
```

If it can access the device, the command will be waiting for input, so talk to your modem by

typing `AT[Enter]`. If your modem is functioning, it will return `OK`. If you get this far, you're ready to attempt a connection to the Net. To return to the command prompt, type the tilde symbol (`~`) and jump to the next section.

If you don't get the expected result, then you may be referencing the wrong device file. Remember `/dev/cua/a` is for serial A/COM 1 and `/dev/cua/b` is for serial B/COM 1. Try again using the alternative.

If you're running Solaris 2.5.1 x86 and your modem is connected to COM 2, you'll need to edit the file `/platform/i86pc/kernel/drv/asy.conf` and uncomment the line that enables COM 2. Then you need to perform a reconfigure boot so Solaris can recognize the port. To achieve this, do the following:

```
# touch /reconfigure
# /usr/sbin/shutdown -y -g0 -i6
```

Once you're back, try talking to the modem again. If you don't get the expected results, jump to the troubleshooting section.

Establishing a connection

Now your system should be in a state where this attempt is going to work the first time, so let's try it. Execute the following command to initiate a connection to your ISP:

```
# /opt/FreePPP/bin/pppd call isp
```

This command will fork in the background while it goes to work. If your modem's speaker is on, you should hear it dialing. By the way, if you haven't run the command yet, it's a good time to do so now.

Given that it's your first time, we should also monitor the `ppp.log`, so run the following:

```
# tail -f /var/log/ppp.log
```

The messages produced are relatively easy to interpret. You'll be able to determine when you've successfully connected, because the log file will contain these messages:

- Remote Login Succeeded—Indicating that authentication succeeded.
- Remote IP address—The IP address of your ISP's end.
- Local IP address—The IP address the ISP gave your system for this PPP session.

Once you see these messages, interrupt the tail command by pressing `[Ctrl]C`.

Now we should test your system's ability to talk to your ISP's DNS server. Run this command:

```
# /usr/sbin/nslookup www.yahoo.com
```

If this command returns output similar to the text below, then you're connected.

```
Server: <ignore this line>
Address: <and this one>
```

```
Name:      www1.yahoo.com
Address:   204.71.200.66
Aliases:   www.yahoo.com
```

If it doesn't, read the section on troubleshooting.

Monitoring your session

PPP also comes with a neat tool to monitor the throughput of your PPP session. Simply run the following command:

```
# /opt/FreePPP/bin/pppstats -w 1
```

The columns of interest are labeled IN and OUT. These figures represent the total number of bytes coming in and going out per second.

As you become an experienced PPP user, it's a good idea to use this tool to measure the performance so you can tune the modem to achieve an optimum configuration. For comparison's sake, our US Robotics 56k v.90 Sportster can peak at 5600 bytes when downloading compressed data. If you do better than this, you're doing extremely well.

Disconnecting

Disconnecting your PPP session is a matter of killing the process ID of the `pppd` daemon. When you establish a connection, the `pppd` daemon's process ID is written to `/etc/ppp/ppp0.pid`. So you can conveniently disconnect by running something like this:

```
# kill `cat /etc/ppp/ppp0.pid`
```

As a first-time user, you should ensure your modem's on-hook (OH) light is turned off, to verify that PPP did the right thing and signaled the modem to hang up. Otherwise, you may receive an unusually large telephone bill next month. Just don't blame us; we warned you.

Troubleshooting

Okay, so it didn't quite work out the way we wanted it to. Let's diagnose the common causes together. The best place to start is the `ppp.log` file. Glance over the messages and see if the problem looks obvious. The following sections explain the common causes of failure.

Authentication failed

Your PPP session failed to authenticate with your ISP. Check the file `/etc/ppp/pap-secrets` and ensure the user name and password corresponds to your ISP account's login name and password.

PPP timed out

Your PPP session failed to negotiate PPP with your ISP. Some ISPs don't support PPP detection. This is where your system dials up and says, "Let's negotiate a PPP session." If the ISP's dial-up systems don't detect this request, your attempt to establish PPP will eventually time out.


You're not totally at a loss, but then the configuration of PPP becomes cumbersome. Un-

fortunately, the required configuration is fairly detailed and can vary; therefore, you need to sift through the available documentation. We suggest you read the `SETUP` file in `/opt/FreePPP`.

Can't talk to Modem (x86 only)

Solaris has not recognized your COM port. Configuring Solaris x86 to recognize modems is a common problem, especially internal modems. Unfortunately, there are too many scenarios to discuss here. If you have access to SunSolve, we suggest you look up SRDB ID 16494. Alternatively visit Celeste's tutorial on Solaris 2.x Modems and Terminals at www.stokely.com/unix.serial.port.resources/modem.html.

Further Information

The FreePPP package contains a good source of documentation if you're experiencing difficulties. Man pages for `pppd`, `chat`, and `pppstats` are in `/opt/FreePPP/man`. The `FAQ`, `SETUP` and `README` files are in `/opt/FreePPP`. 

All about SunPC cards

by Asim Zuberi

One fine day, my manager asked how much I knew about SunPC cards. I shook my head and told him I'd never heard of such a beast. He asked me to jump on it right away and gather information to find out if they were worth getting for our Sun workstations.

I did some research and found out that Sun Microsystems has a hardware product called *SunPC*. The SunPC card comes with SunPC software, which provides PC hardware environment emulation on SPARC workstations running Solaris. The SunPC card is just like a SCSI card that takes one of the SBus slots of your workstation, allowing you to run DOS and MS Windows (3.11 and 95) along with your Solaris applications on your worksta-

tion. So, what does it buy us? Well, first of all, it's eliminated the need for a stand-alone PC, and, second, it allows you to run Microsoft products on hardware that's very stable and reliable.

Moreover, the SunPC card has its own 133 MHz AMD 586 processor, which is used only by Windows 95 stuff; thus it leaves the SPARC processor alone, and the SunPC card doesn't affect your workstation performance. SunPC software lets you mount NFS-based directories as if they were extended drives on a PC. It has the ability to support display resolution of either 800- by 600-dpi or 1024- by 768-dpi.

After hacking around, I gathered the above information and presented it to my manager, who was amazed with its capabilities. We

decided to purchase 14 SunPC cards and plug them into our desktops. This article details the steps I went through to install and configure the SunPC card, the software, and Windows 95 on the Sun workstation (running Solaris 2.6).

Installation of the SunPC card

First, you need to shut down your system to install the SunPC card; you must also have root access to install the SunPC card and its software by typing `# init 0`.

This will bring the machine to the Ok prompt. You can now power-off your system, remove the cover of your box, find an empty SBus slot, and securely install the SunPC card. Put the cover back on, make all the necessary connections, and restore power to your system. The moment you power-on the system, it will start booting. We want to reconfigure the device before it comes to a multi-user mode, so halt the system by pressing [STOP]A. Now the Ok prompt re-appears on your screen. At the Ok prompt, type `ok boot`.

Installation of the SunPC software

Now, let the system boot all the way to multi-user mode. Once the system is online, log in as root (if you feel comfortable in full screen mode, I'd suggest you use it and don't run X-windows), put the SunPC (4.2) CD-ROM in the CD-ROM drive, and wait until the volume manager mounts it. Once it's mounted (you can check by typing `df -kl`), type the following at your root prompt:

```
# cd /cdrom/sunpc_4_2/Product (change directory to the CD-ROM mount point)
```

```
# pkgadd -d `pwd` SUNWsunpc
```

It's easiest to just accept the default choices to the questions asked by the `pkgadd` command. The `SUNWsunpc` package will be installed in the `/opt` directory on your system.

Configuration of SunPC Software

Once the `SUNWsunpc` package has been installed properly, exit from the session you're in (that is, Open windows or CDE). Using the command line login, log in as root, and do the following (Note that it's important to run the `sunpc_install` script from the command line login with no X-windows):

```
# cd /opt/SUNWsunpc/bin
```

```
# ./sunpc_install
```

The `sunpc_install` program will install the SunPC software, and a completion message will be displayed when it's done.

Halt the system when the `sunpc_install` program is finished, as follows `# init 0`. The system will come again to the okay prompt. Type `ok boot -r` to restart your workstation.

Once the system is back online, log in as a user, *not* as root. We do this because we want to set up the Win95 stuff in the user's home directory. Edit either your `.profile` file (if he's using the korn and bourne shell) or `.login` file (if he's using the C shell), and add the `/opt/SUNWsunpc/bin` directory to your PATH variable. Personally, I edited the `/etc/profile` and modified the PATH there, so it's set once and for all. To get it activated, either execute the profile file or log out and log back in.

Now, the final step is to configure the SunPC software. Before we go too far, make sure that you have at least 300 MB of free disk space available in your home directory. If not, then you might want to think of an alternate way to get that amount of disk space.

Setting up your disks

Enter the `sunpc` command at the command line. In a few seconds the SunPC 4.2 window will appear on your screen, as shown in **Figure A**. This window will look just like your terminal window, except that it has some menu options. Since this is the first time you've run the SunPC software, it will

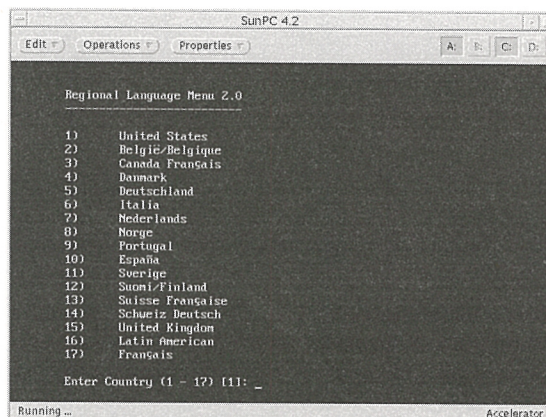


Figure A: This screen appears when you start the installation utility.

create a `pc` directory in your home directory and a run control `sunpc` file: `.sunpcrc`. The Regional Language Menu 2.0 will be displayed. Choose number 1 if you want to select the United States.

Right-click the Properties button at the top of the SunPC 4.2 window, and the Properties menu will be displayed. Select the Hard Disk option from the Properties menu. The SunPC: Drive Properties window will be displayed, as shown in **Figure B**. Click the Hard Disks button, and the Hard Disk Editor window will open, as shown in **Figure C**.

Enter the drive name in the New Disk Name field. Actually, this is the filename that will be created in the `pc` directory. You can name it whatever you want. (Just to make it logical, I called mine `C.win95`.) In the Disk Size field, you can enter the size of the file you want. (I would recommend 128 MB.)

Once you've entered the disk name (or filename) and have set its size, then click Create Disk. The new disk name will pop up in the list box. Highlight the disk name in the list box

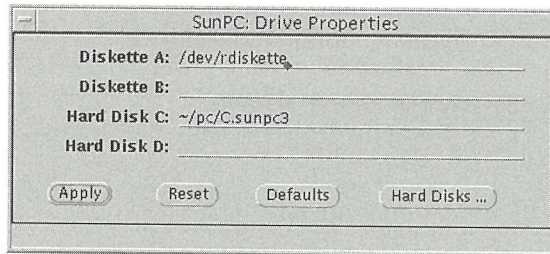


Figure B: This is the hard disk Properties screen.

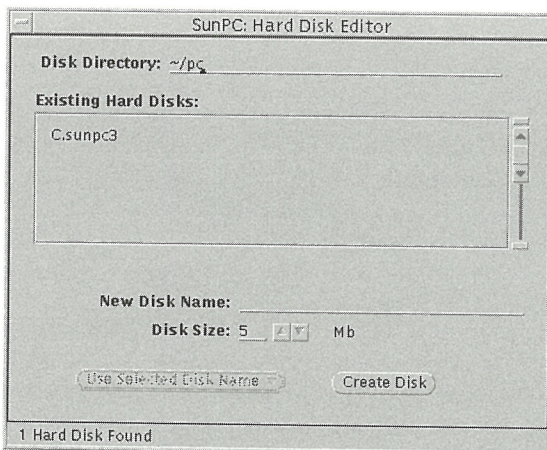


Figure C: This window is where we change our hard disk properties.

and click Use Selected Disk Name. A drop-down menu will be displayed. Select Drive C, and it will prompt to reboot the SunPC card. Click on Okay To Reboot. Once it's back, it will display the Regional Language Menu 2.0 again. Choose 1 for United States.

Select the Drive option again, and repeat the previous steps to create another drive (or another filename) for Office 97, and set its size to 128 MB, as well. (I called it `C.Off97`) Set the disk name as Drive D. This way, Office 97 files won't get installed in your `C.win95` filename, and you'll still have room to install other applications on your `C.win95` drive. (Remember, you've only allocated 128 MB of disk space to it.) Reboot your SunPC card, so that the changes can take effect. Note that only your SunPC card needs to be rebooted and not your workstation.

Managing your memory

By now, we've created two 128-MB files to install Win95 and Off97 in the default location (`~/pc`). Next, let's set the memory size. Your system will share its physical memory with the SunPC card. If your system has 64 MB RAM, you can assign 16 MB (the maximum memory size that Sun supports on SunPC cards) to the SunPC card; if your system has 32 MB RAM, then it will only share 8 MB RAM with the SunPC card. With 8 MB RAM, the SunPC card runs fine, but it's a bit slow.

Now, we can configure our memory size. Make sure that your SunPC window is up and running. Right-click on the Properties menu and select the Miscellaneous command from the pop-up menu. You'll see the window shown in **Figure D**.

In the Memory Size field, type in either 8 or 16, and click Apply to return the SunPC Window. It will prompt you to again reboot the SunPC card.

Installing Windows 95

The next step is to install Windows 95. To do this, we need to use the `prep` utility. This is a batch file that copies the necessary drivers and DOS files from the read-only F drive to your local C drive (for example, `C.win95`). You must run the `prep` command before installing Windows 95; see **Figure E**.

```
C:\> f:\drivers\win95\prep
```


Once it's finished, close the SunPC window. It will prompt you to confirm Quit SunPC Window.

You shouldn't be logged in as root to install Windows 95. Insert the Windows 95 CD in your CD-ROM drive and wait until it's mounted by the volume manager. Once it's mounted, run your `sunpc` command `$ sunpc`.

At the C prompt, type `C:\> r:\` (where *R* refers to the CD-ROM drive) `R:\> cd \cdrom\cdrom0`. Now, locate the command `setup.exe`, which resides on the Win95 CD-ROM that you just put it in. It's usually under `\cdrom\setup.exe`. Once you locate it, switch to this directory and type `R:\> setup.exe c:\bsetup.inf`.

First, it will run the Microsoft Scan Disk, and then will start loading Windows 95 onto your system, which may take several minutes. You'll then see some Windows 95 messages displayed on the screen.

During the installation of Windows 95, the Microsoft License Agreement will pop up; click Okay. Your black arrow insertion point won't work in the Windows 95 window, so you'll have to enable the Windows 95 white arrow insertion point. In order to do this, press [Meta]M. This will make the white cursor active and the black cursor inactive. Now if you want to switch back and forth, [Meta]M is the trick.

Configuring long filenames within Windows 95

By default, Windows 95 doesn't support long filenames, and your screen display is set to 1024-by-768. In order to get long filenames, some additional drivers need to be installed. Here are the instructions on how to do that. (Make sure your SunPC CD is in the CD-ROM drive, and that it's mounted by the volume manager.) Run Win95, click Start, and choose Run. The Run window will pop up; then, type in the Run window `f:\drivers\win95\setup\setup`.

Setup will install the following files from the SunPC CD and will prompt you to insert the Win95 CD. Replace your SunPC CD with the Win95 CD and then click OK, so that the setup finishes loading the rest of the drivers.

(Note: These files reside under `/cdrom/sunpc_4_2/Product/SUNWsunpc/SUNWsunpc/reloc/SUNWsunpc/dos6/`

`drivers/win95`.) Here's a list of files that will be installed from the SunPC CD:

```
spyg7x9.fm
sunfsd.vxd
sunnp.dll
sunpc.drvc
sunwmbox.vxd
sunwndis.vxd
supervga.drvc
```

If, for some reason, the Windows 95 setup couldn't locate the files on the SunPC CD-ROM, copy these files manually to your home directory. You can then have setup look in your home directory instead of the CD.

Once setup is done installing the drivers both from SunPC CD and Windows 95 CD, it will prompt you for a reboot of Win95. Don't select the reboot then; instead, go into the display settings and set 1024-by-768 with 256 colors and reboot your Win95.

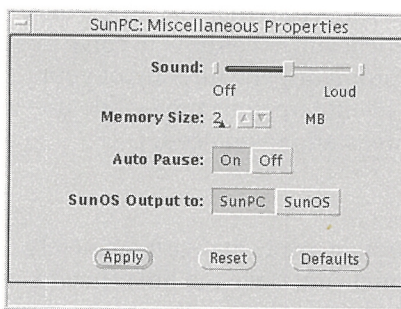


Figure D: Set miscellaneous properties to your values here.

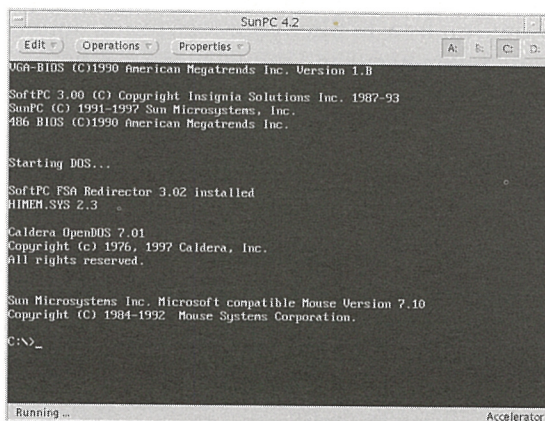



Figure E: Using the SunPC DOS window from our workstation will look like this.

Now, you should notice that when Win95 comes back, its window size is bigger (1024-by-768). Log in as a user in Win95 and bring up the Windows Explorer. Under the Tools menu, select Map Network Drive. In the window displayed on the screen, choose any drive

letter and the path of your workstation—usually your home directory (for example, `\\home\username\`). Once this is set, you can access the long filenames within this newly created drive letter. Enjoy using Windows 95 on your Sun Workstation! 

Watching your logs

by Lance Spitzner

Logs are a critical asset to successfully running our systems. They tell us what is and what isn't happening. However, logs can be extremely copious, quickly overwhelming us with information. Pretty soon they become useless files that just fill up disk space. This article will cover how to solve this problem by automating the filtering of your logs, thereby freeing up your time while presenting the information you need.

Filtering

Logs are incredible assets that, unfortunately, are often ignored. You have too little time to review too much information. Wouldn't it be nice to automate the process—to create a process that reviews the logs for you, and then notifies you with only the information you need? Well, we're going to do just that. We'll show you how to filter your logs for the information you need, and implement a notification system.

The first part of the article will cover developing a plan on what you want to filter and what you want to be notified of. The second half will be on implementing the filter. For this article, we'll be using SWATCH as a filter, written by Todd Atkins. We'll also be using sendmail logs as an example of the filtering process. However, you can apply these guidelines to any type of logging.

Where to begin

The best place to start is with a plan. There are three steps to planning for automated logging.

The first step is to define what you want to know. What information do you need from your system logs? The second step is to determine which logs contain that information. The third step is to identify the trigger, or determine what defines the critical information.

For example, let's say you're concerned about the security of your sendmail. Specifically, you want to know if someone attempts to use your mail server as a spam relay. You also want to know if anyone is attempting to gain unauthorized information with SMTP commands, such as `expn`. We've completed the first step, which is identifying what we want to know.

The next step is identifying the source, or which log contains this information. The best place to find that is `/etc/syslog.conf`. This configuration file will show you what information is logged where. For mail, we see that all mail information is logged to `/var/log/syslog`.

```
mail.debug   ifdef('LOGHOST', /var/log/  
syslog, @loghost)
```

The last step is defining the trigger. What specific entries in the logs define the information we're looking for? In the case of sendmail, we're looking for two triggers:

1. A trigger showing unauthorized IP addresses attempting to use our mail server as a mail relay.
2. A trigger showing that someone is attempting to use the expand command, which we've turned off.

The best way to define the trigger is to re-create the incident while monitoring the log with `/usr/bin/tail -f`. If you can't do this on a production system, find a lab system you can replicate the trigger on.

First, let's re-create the incident for the first trigger—unauthorized use of our system as a mail relay. From an unauthorized IP address, attempt to use your mail server as a relay. With `/usr/bin/tail -f` you see the log entry in `/var/log/syslog`, shown in [Listing A](#).

Here we see the error message of someone at `moo.com` attempting to relay email, usually a sign of spam. This is the trigger for unauthorized mail relay. Notice that the error also includes the IP address, verifying the domain.

Now, let's re-create the second trigger—unauthorized use of the `expn` command. Telnet to the SMTP port and execute `expn`. Meanwhile, monitor the `/var/log/syslog` with `tail -f`, as shown in [Listing B](#).

There we see the error message of someone at `moo.com` attempting to expand the user name `bsmith`. This is the trigger if anyone attempts to exploit the `expn` command. Notice that the error also includes the IP address, verifying the domain.

We've now completed the three steps in planning for automated log filtering. We first identified the information important to us, unauthorized attempts to use our mail server as a mail relay and use of the `expn` command.

We then identified the logs that contain this information. Last, we identified the triggers for this information by re-creating the incidents. We're now ready to build our automated filter.

SWATCH

SWATCH, "The Simple WATCHer and filter," is a Perl program developed by Todd Atkins that monitors your logs in real time. SWATCH monitors your logs for specific triggers, and when those triggers are matched, SWATCH notifies you in a pre-determined manner. In our case, we're going to implement SWATCH to alert us whenever someone is messing with our sendmail.

The program is extremely simple to install. Since this is a Perl program, no compiling is involved. SWATCH comes with a useful install script that will copy all the libraries, man pages, and Perl files to their respective directories. Once installation is complete, all that's left to do is create a configuration file

and then launch the program. You can download SWATCH at <ftp://ftp.stanford.edu/general/security-tools/swatch>.

The configuration file, called `swatchrc`, is the heart of the SWATCH program. This text file tells SWATCH which logs to monitor, which triggers to look for, and what to do if triggered. SWATCH works by looking for regular expressions that match the triggers defined in `swatchrc`. When it finds a match, it executes the notification procedure defined in `swatchrc`. SWATCH monitors the files in real time, using `/usr/bin/tail -f`.

We'll now create a `swatchrc` file for the sendmail logging we discussed above. The goal is to have sendmail email us whenever someone is messing with our email system. We defined this earlier as anyone attempting unauthorized mail relay or the `expn` command.

The syntax of a `swatchrc` file is as follows. It consists of four tab-delimited fields. The first two fields are required, but the last two fields are optional. The first field is `/pattern/pattern/` where *pattern* is a regular expression that SWATCH is looking for. This is our trigger. The second field is `Action,action-`, where *action* is what to do if the pattern is matched. SWATCH has various options for actions, including email, paging, or executing any file you select.

The third field (which is optional) is a time interval defined as `HH:MM:SS`. *HH* is for hours, *MM* for minutes, and *SS* for seconds. This time interval is the amount of time SWATCH will ignore identical matched patterns that

Listing A: Trigger for anyone attempting unauthorized mail relay from your sendmail server

```
Oct 3 14:48:51 homer sendmail[6704]: OAA06704:
    ruleset=check_rcpt,
    arg1=bsmith@domain.com, relay=foo@moo.com
    [206.54.252.1],
    reject=550 bsmith@domain.com... Relaying denied
```

Listing B: Trigger for anyone attempting to utilize the expn command on your sendmail server

```
Oct 2 20:28:37 homer sendmail[5453]: NOQUEUE:
    foo@moo.com
    [206.54.252.1]: expn bsmith [rejected]
```


repeat themselves. For example, if you define this period as five minutes, SWATCH will only report one identical matched pattern over that time period, even though it might have matched twenty identical entries.

The fourth field (required if you're using the third field) is a timestamp, defined as `start:length`. This defines the location and length of the timestamp in the notification message.

For our sendmail example, we want to create a `swatchrc` file that looks for patterns matching our two triggers shown in [Listings A](#) and [B](#). When SWATCH matches either of these patterns, we want it to notify via email at `abuse@ourcompany.com` and to include the matched pattern in the email message. However, we have to be careful not to be flooded with warnings. For example, if someone attempts to relay off us with 1,000 emails a minute, we'd be overwhelmed with notifications. So, we'll set a time interval of five minutes. Regardless of how many identical patterns are matched in a five-minute period, we'll receive only one warning. Our `swatchrc` file would look as follows:

```
/Relaying denied!expn/ echo=normal,  
mail=abuse@company.net 5:00 0:16
```

The first field has `/Relaying denied!expn/`. If SWATCH matches either pattern in the regular expression, it will send an alert. The first pattern, `Relaying denied`, is found in trigger #1, or [Listing A](#). This log is the result of someone attempting an unauthorized mail relay. The pattern `expn` is found in trigger #2, or [Listing B](#), and is the result of someone attempting to use the `expn` command. You'll find both expressions in the triggers we covered in the first part of the article.

The second field states `echo=normal`, `mail= abuse@company.net`. This field instructs

the program to email a warning to `abuse@ourcompany.net`, and to echo the matched log entry.


The third and fourth fields (which are optional) have `5:00 0:16`. This tells the program not to repeat any warning for identical patterns matched within five minutes. The last field states the location and length of the timestamp.

We now have a properly configured `swatchrc` file. The last step is starting SWATCH itself. SWATCH can be launched with a variety of options. However, we'll launch with the following syntax.

```
/usr/local/bin/SWATCH -c /var/log/syslogrc -t  
/var/adm/syslog &
```

The `-c` option points to the configuration file, and the `-t` option monitors the log file in real time. The `&` runs the SWATCH in the background. Once launched, SWATCH forks a child, so SWATCH will be running as two processes. Be sure to kill both processes in any stop/start scripts you create. That's it. Your sendmail logs will be automatically filtered. Whenever someone messes with your sendmail system, you'll be instantly notified via email, with the matched trigger in the log included, as we've shown in [Listings A](#) and [B](#).

Conclusion

Logs are powerful tools, yet they can easily overwhelm us with data. When this happens, we start ignoring this valuable resource because we don't have time to scan through megabytes of data. Automating the filtering of such logs solves the problem. These automated filters do the work for us, alerting us in real time with the information we need. We hope this article has given you some good ideas on how to automate the filtering of your log files. 

Coming up...

- OpenBoot programming
- *Inside Solaris 1998 Index*
- Glimpse and Webglimpse (Web search tools)
- Text processing and lexical analysis

PING THE SOLARIS DUDE: SOLARIS Q & A

by Robert Owen Thomas

This month begins a new look for the column. Along with multiple and varied questions, I've added a "Tool of the month" section at the end of the column. So keep those questions coming to robt@cymru.com!

Is the process still alive?

From within a program, you may wish to determine if a corollary process is running. Within a shell script or Perl, this can be accomplished with the use of `ps(1)`. However, it's less than efficient to use `popen()` or (the infamous and evil) `system()` within a C program. **Listing A** shows a simple solution using, of all things, `kill()`:

By simply sending a signal 0 to the process, we can determine if the process still exists. Remember that permissions still count with signals, and only root can send a signal to any process. This snippet of code can be cut and pasted into any application for which process monitoring is necessary.

What is a memory leak?

A memory leak results when allocated memory (through the use of `malloc()`, `calloc()`, and friends) isn't freed via `free()`. While many programs have no need to dynamically allocate memory, long-lived programs—including the Solaris operating system—do need to dynamically allocate (and de-allocate) memory.

A program with a memory leak will consume memory over time. The easiest way to see this growth is through the use of the `ps(1)` command in `/usr/ucb`. Watch the SZ, or size, column for a period of time. If it grows continually and becomes quite large, you have an application with either huge memory requirements or, more likely, a memory leak. You should also check swap (with `swap -s`) and the output of `vmstat(1M)` to see if available system memory is decreasing. If you have the Solaris Dude's `memtool` tool, you can monitor the stack and heap of a potentially leaking process.

Listing A: Use the `kill()` function to see if a process is still alive.

```
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>
#include <errno.h>

extern int errno;

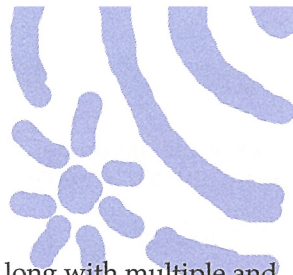
int
main(int argc, char **argv)
{
    int ret_val;
    pid_t mypid;

    mypid = atoi(argv[1]);

    ret_val = kill(mypid, 0);

    if (0 != ret_val) {
        /* Look for ESRCH */
        if (3 == errno) {
            fprintf(stderr, "Proc %d not
                ↪ alive.\n", mypid);
        } else {
            perror("kill");
        }
    }

    fprintf(stdout, "Proc %d still
        ↪ active.\n", mypid);
    exit(0);
}
```



Inside Solaris™

Tip and techniques for users of Sun 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.

Editorial

Editor.....Garrett Suhm
Assistant EditorJill Suhm
Copy Editors.....Rachel Krayer
Christy Flanders
Taryn Chase

Contributing EditorsRichard Auletta
Peter Marelas
Lance Spitzner
Robert Owen Thomas
Asim Zuberi

Print Designer, Cover and Content DesignLance Teitsworth

General ManagerJerry Weissberg
Editor-in-Chief.....Joan Hill
Editorial DirectorMichael Stephens
Managing Editor.....Kent Michels
Circulation Manager.....Jeff Marcus
Print Design Manager.....Charles V. Buechel
VP of Operations and Fulfillment.....Michael Springer

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@zsjournals.com.

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

Postmaster

Periodicals postage paid in Louisville, KY.

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.

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.

Price

Domestic\$129/yr (\$11.00 each)
Outside US\$149/yr (\$13.00 each)

Our Canadian GST # is: R140496720.

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost \$11.00 each, \$13.00 outside the US. You can pay with MasterCard, VISA, Discover, or American Express.

ZD Journals publishes a full range of journals designed to help you work more efficiently with your software. To subscribe to one or more of these journals, call Customer Relations at (800) 223-8720.

SunSoft Technical Support
(800) 786-7638

PERIODICALS MAIL


C:7661977 0002096 04/00

WNSHIP MI 48035-4218

1111111111

Please include account number from label with any correspondence.

If you have access to the source code, you can avoid a memory leak with a couple of tips. First, always pair up each `malloc()` with a `free()`. Never leave a subroutine that has a `malloc()` without calling `free()`. Second, when locally scoping a dynamically-sized variable, use `alloca()`, which allocates space on the stack. Note that this space will be freed upon return from the subroutine.

Keep in mind that, in SVR4, memory released by `free()` isn't returned to the kernel free list. When freed, the memory is returned to the heap of the calling process. From here it can be used for future memory allocation requests. While not a true memory leak, this can obviously cause a slow reduction in available system memory. 


TOOL OF THE MONTH

Network top

The tool of the month is `ntop`, or network top. Many of you are undoubtedly familiar with the ubiquitous `top` command. Wouldn't it be nice to monitor network activity as easily as we monitor system activity? Now you can!

`Ntop` displays the output in either a curses window or a Web interface. It maintains state information for all network activity on a local host. With the Web interface, `ntop` can even display distribution statistics, such as the per-

centage of TCP versus UDP traffic. This is a very handy tool for system admins, and can be quite helpful when debugging thorny network issues. With the gathered statistics, it's certainly easier to tune performance.

You can obtain `ntop` at the following URL: www-serra.unipi.it/~ntop/. The Solaris Dude makes no warranty regarding this tool or its use. However, should you encounter any problems porting or using the tool, don't hesitate to drop me a line. 

About our contributors

Richard Auletta started dialing up computers when 300 baud was considered fast. He can be reached at rauletta@uswest.net.

Peter Marelas is based in Melbourne, Australia and currently works for the Fulcrum Consulting Group as a senior consultant. He specializes in Solaris, High Availability, Security, and Networking. He remains a long-standing member of the Solaris community, and frequently participates in the transfer of knowledge and development of new tools. You can reach Peter via email at maral@phaseone.com.au.

Lance Spitzner enjoys learning by blowing up his UNIX systems at home. Before this, he was an Officer in the Rapid Deployment Force, where he blew up things of a different

nature. You can reach him at lspitzner@enteract.com or www.enteract.com/~lspitz.

Robert Owen Thomas is an aspiring blues guitarist earning his living as a UNIX and networking consultant. He can be contacted through email at robt@cymru.com, or visit his Web site at www.cymru.com/~robt.

Asim Zuberi received his MS degree in mechanical engineering in May 1993, at the New Jersey Institute of Technology, NJ and works now at Collective Technologies (a Pencom Company). Currently, he's onsite at Lucent Technologies, one of CT's clients. He started with UNIX in 1992, worked with Sun, SGI, and Linux, and has been heavily involved in Solaris administration since 1996. You may reach him at asim@colltech.com.