ZD JOURNALS

May 1999 • $9.00
Vol. 5 No. 5
www.zdjournals.com/sun

Inside Solaris™

Tips & Techniques for users of Sun Solaris

xDSL and cable modems

by Richard Auletta

With the wide spread availability of xDSL and cable modems, low-cost megabit direct access Internet connections are now available for the home and small office. Unfortunately, xDSL and cable modem providers rarely support Solaris officially. This article describes how to connect your Solaris system and home or small office network to the Internet with xDSL and cable modems, even when Solaris isn't officially supported.

xDSL

A digital subscriber loop (DSL) modem offers a high-speed, full-time connection to an ISP and the Internet over standard copper phone lines. Current xDSL modems, the *x* standing for a range of DSL protocols, support data rates up to 7,000 Kb/sec downstream and 1,000 Kb/sec upstream.

xDSL accomplishes this high-speed magic by only making use of the copper wires to the telephone central office (CO), as shown in **Figure A**. The DSL signal is split off at the CO, effectively leaving the phone system, and jumping directly to a data switch and then to an ISP. Because the voice switch at the CO isn't used for the xDSL connection, your plain old telephone service (POTS) will operate simultaneously over the same phone line.

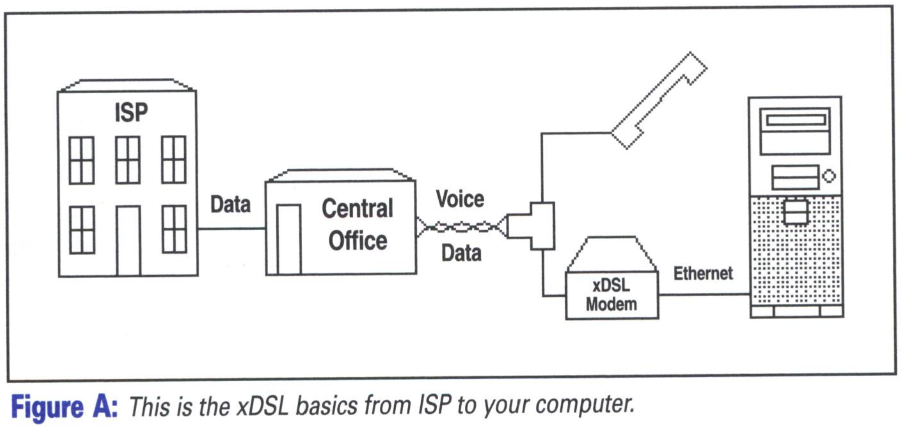

Figure A: *This is the xDSL basics from ISP to your computer.*

Cable modems

Cable modems, based on the Data Over Cable Service Interface Standard (DOCSIS), operate over your local cable television system and provide peak data rates on the order of 1,000 Kb/sec. Like an xDSL modem, the connection is full-time. The main difference is that the bandwidth of a cable modem subnet is shared by all users on the network segment, typically 500 to 5,000 homes. The bandwidth of the shared connection is typically 27 Mb/sec to 36 Mb/sec downstream and 500 Kb/sec to 10 Mb/sec upstream. The effective data rate will depend on the network traffic and the number of active users.

xDSL or cable modem?

If your area has both xDSL and cable modem service, you'll want to consider several issues before deciding which is right for you. The most important fact to remember is that

In this issue:

ZD ZIFF-DAVIS a SOFTBANK company

Internet download speeds are typically constrained more by the remote server and network capacity than your local connection, making very high local bandwidths less important than they might appear at first. Some cable modem operators limit the bandwidth available to each cable modem.

A distinct advantage of xDSL over cable modems is the more commonly available option to choose from different ISPs. This allows selecting an ISP that offers the services you need. xDSL also allows a business to become an ISP and extend its internal network directly to the homes of telecommuters without the need to cross the Internet.

Basic configuration

Unlike the more familiar dial-up PPP services that use a serial connection to make the final connection from computer to modem, xDSL and cable modems use a 10Base-T or 100Base-T Ethernet interface for the local connection to your computer or network. As with a dial-up PPP connection, you'll still have an Internet service provider that will supply you with a static or dynamic IP address, POP mail accounts, and other services. But unlike configuring a dial-up PPP connection, which can be difficult, configuring Solaris to work with xDSL and cable modems can be as straightforward as configuring Solaris on any Ethernet-based network.

With a single computer, and no home network, the xDSL or cable modem connects directly to the Ethernet port on your Solaris system, as shown in **Figure B**. The modem will come with the appropriate Ethernet cable, typically a cross-over cable, to support the direct connection. If you have a home network, you can also connect the modem to your hub or second Ethernet interface. This configuration is described later in this article.

Static IP

If you arranged for a static IP address from your ISP, then setting up your Solaris system couldn't be easier. Simply run the command sys-unconfig as root. This will cause the system to shut down and reboot. When your system reboots, you'll be prompted for the fully qualified hostname, IP address, netmask, and other system particulars needed to configure your system on a network. If you're installing Solaris, the installation interface will prompt you for this information.

Create an /etc/resolv.conf file, as shown in the following code snippet:

```
nameserver 129.176.20.21
nameserver 129.176.20.119
```

Don't forget to edit your /etc/nsswitch.conf file with

```
#Add dns to hosts line
hosts: files dns
```

to enable the DNS `resolver` routines to look in /etc/hosts, and then to use DNS to map a name to an IP address. There's no need to reboot after making these changes to resolv.conf and nsswitch.conf. Your ISP will also supply the IP addresses for domain name servers (DNS).

You may also find that your ISP doesn't supply routing information and `routed` won't automatically set up your default route. You will know this if you can't reach remote machines even after you've set up your connection and enabled DNS. Also, if the command `netstat -rn` doesn't list a default route after rebooting, then create an /etc/defaultrouter file with the

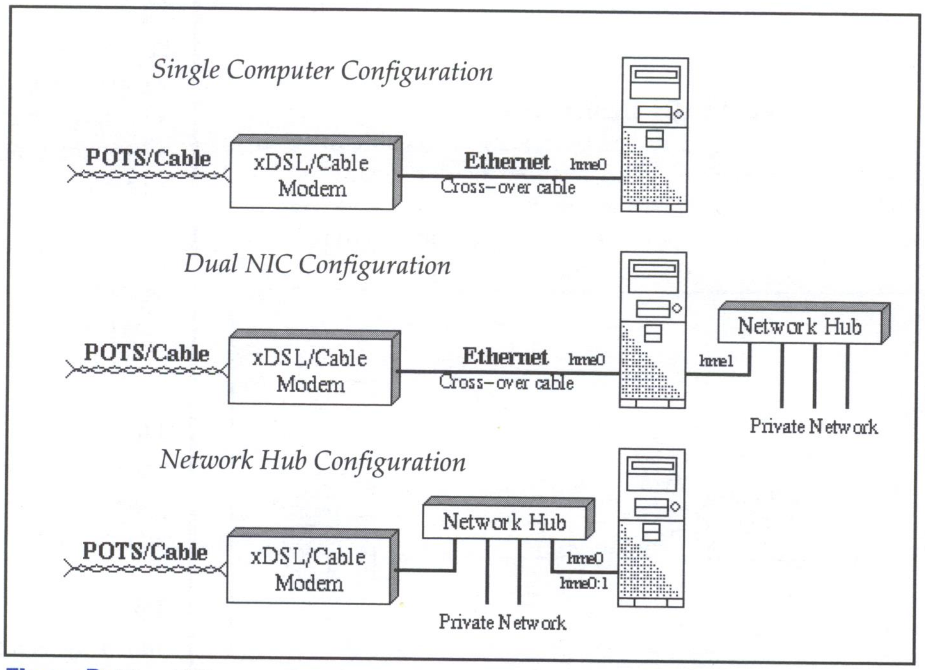

Figure B: *The xDSL network connects directly to your Ethernet network.*

IP address of the gateway machine supplied by your ISP and reboot. You can add the route manually with the command `route add default gateway_ip_address`. Solaris will automatically set the default route at boot once /etc/defaultrouter exists. The default route gateway address is typically the same IP address as your machine's IP address, with a final octet of 254 or 1.

Dynamic IP

If your ISP uses dynamic IP addresses, you can still use your Solaris system. Solaris 2.6 and 2.7 will attempt to acquire a DHCP-assigned IP address when the files /etc/dhcp.<interface> and /etc/hostname.<interface> both exist. The <interface> is the name of the Ethernet adapter that's to set its IP address via DHCP.

The name of the network interface will typically be hme0 on UltraSparcs or le0 on newer Sparc machines. On Solaris x86 systems, the name will be more variable, changing for each brand of network interface card (NIC). In both cases, search /var/adm/message files for the network to discover the interface name. Typical names are nei0 or elxl0. If you have more than one interface, the 0 will be replaced by a 1, 2, etc.

Both /etc/dhcp.hme0 and /etc/hostname.hme0 can be empty. You can create them with the touch command: `touch /etc/dhcp.hme0` and `touch /etc/hostname.hme0`. At the next reboot, your Solaris system will attempt to acquire a dynamic IP address early in the boot process.

Your ISP's DHCP server should supply DNS servers, default route, hostname, and netmask, and if supplied, Solaris will set these for you. If the ISP's DHCP server doesn't supply these fields in the DHCP response, then you'll need to set them manually as described in the previous section on a static IP address. Typically, these additional fields aren't present in the DHCP response. Otherwise, you'll only need to edit /etc/nsswitch.conf, as explained in the previous section, to enable the `resolver` routines to use both files and DNS for hostname lookup.

The Ethernet interface is configured by three shell scripts in /etc/init.d that run during the boot process: rootusr, inetinit, and inetsvc. These shell scripts are well commented and can be useful in understanding how Solaris calls `ifconfig` to configure the interface and system files for DHCP. You can get additional information about `ifconfig` from its man page.

Home network

Unlike low-speed POTS modems, high-speed xDSLs and cable modems make it practical for multiple computers to share a single DSL or cable modem connection. The simplest option is to arrange for multiple dynamic or static IP addresses from your ISP. Many xDSL ISPs offer this option for a small, extra fee, and some cable systems simply allow multiple dynamic IP addresses to be acquired. Of course, more traditional services are available, such as domain hosting and the routing of a registered private domain.

Basic equipment setup couldn't be easier when working ISP-assigned IP addresses. Simply connect your xDSL or cable modem to your network hub and configure each of your machines for either static or dynamic IP address. Don't forget to secure your machines against unauthorized access, as each is now directly accessible from the Internet.

Building the home network

An alternative to multiple ISP-supplied IP addresses is to create a private network using an IP address space reserved in RFC 1918 Address Allocation for Private Internets for non-routing domains. One such IP address space is 192.168.0. This IP address space allows the creation of a local network with IP addresses 192.168.0.1 to 192.168.0.254 that will never conflict with an address on the Internet.

To create your home network, simply assign IP addresses and names of your own choosing to each of your machines from the above address space. Add the IP addresses you have assigned to each of your systems on your network to the /etc/hosts on your Solaris systems. Running a local domain name server won't be covered here, as it's beyond what would be needed for a small network.

To configure your Solaris system to be on your private network, simply run the command `sys-unconfig` and enter your local network name and IP address of the system when the machine reboots. Likewise, use the network icon in the control panel under Windows to configure your Windows systems.

You'll also need to create a lmhosts.txt file to allow reference by name of the other machines on the network from your Windows-based machines. Create a file called lmhosts.txt in C:\WINNT\System32 and load it using the Windows network GUI in the control panel with the sequence control panel → network → protocol → tcp/ip protocol → WINS Addressing. Enable LMHOSTS lookup and import the lmhosts.txt file in the GUI and reboot. The following code is a sample of the lmhosts.txt file:

```
192.168.0.10  tatung
192.168.0.11  dell
192.168.0.12  hp
```

Connecting to the Internet

At this point, you should have an internal network and should be able to ping and telnet to machines on your private network using the names you assigned and placed in /etc/hosts or lmhosts.txt. The next step is to connect your internal network to the external network of the xDSL or cable modem.

If you have two interface cards, you need to configure the second interface to connect to your xDSL or cable modem. Assuming hme0 is your private network, create an /etc/hostname.hme1 containing the hostname of the IP address to be assigned to this interface. The IP address name pair must be in the /etc/hosts file.

If you'll be using DHCP to set the IP address for the second interface, create both an /etc/hostname.hme1 and /etc/dhcp.hme1. You can also bring this interface up without rebooting using the `ifconfig` command. Remember, your NIC interface name will only be hme1 if you're on an UltraSparc. Check your /var/adm/messages for the Ethernet interface's name.

If you don't have two NICs in your Solaris system, you can still connect to both the local private network and your xDSL Internet connection by connecting the xDSL or cable modem to your Ethernet hub or switch. Instead of using two physical interfaces, you can create a logical interface on a single physical interface with its own IP address.

Create an /etc/hostname.hme0:1 file with the hostname of the interface. Remember to include the IP address hostname pair in /etc/hosts. If you want to use DHCP on this logical interface, then also touch an empty /etc/dhcp.hme0:1. Finish the configuration by setting up DNS in /etc/resolv.conf and /etc/nsswitch.conf and configuring your default route as described earlier.

The advantage of a logical interface is avoiding the high cost of NIC cards for Sparc systems, which typically cost upwards of $500. For x86-based systems, a second PCI NIC should be well under $50 retail and will keep xDSL and cable modem traffic off your local network.

Proxies and NAT

The next step is to make the Internet accessible to the other machines on your private network. There are two solutions. First, set up a proxy server that will proxy ftp, telnet, and pop3 services. An alternative is IP network address translation (NAT) that will map the private IP addresses onto a public IP address. While proxy servers can be simpler to configure, NAT will make access to the Internet transparent to the machines on your private network without special configuration of Web browsers or special commands for telnet and ftp.

While there are many commercial packages that provide proxy or NAT service, Ip-filter is a noncommercial firewall and NAT package that runs on a range of UNIX platforms, including Solaris.

Ip-filter compiles and installs easily with just two commands: `make solaris` and `make package` (as root). The following code is a sample /etc/opt/ipnat.conf file to enable network address translation for an internal private network 192.168.22 on hme0 to the public interface 133.45.266.12:

```
map hme0 192.168.22.0/24 -> 133.45.226.12/32
➥proxy port ftp ftp/tcp
map hme0 192.168.22.0/24 -> 133.45.226.12/32
➥portmap tcp/udp 40000:60000
map hme0 192.168.22.0/24 -> 133.45.226.12/32
```

Ip-filter will also act as a firewall, but you can initially leave /etc/opt/ipf.conf empty and configure Ip-filter's firewall capabilities later.

Since you may have a default router file that disables IP forwarding in /etc/init.d/inetinit, you'll need to enable IP forwarding with the command `ndd -set /dev/tcp ip_forwarding 1`. You can add this to the end of /etc/init.d/inetinit to enable it each time the system boots. Either reboot or start Ip-fiter by executing the command `/opt/init.d/ipfboot start`. To finish the installation, make the machine running Ip-filter the default gateway on the other computers on your private network. Also, update the DNS name service to point to the name servers your ISP supplied.

Ip-filter is a kernel module, and any bugs will typically result in a kernel panic, hung system, or other mysterious behavior. We experienced spontaneous reboots after installing Ip-filter 3.2.10 until we applied the most recent kernel patch for Solaris 2.6. We also noted that the Internet interface must be on the non-logical interface hme0, and the private network should be on the logical interface hme0:1 for network address translation to work correctly.

Alternatives

You might decide to use Windows as your gateway machine. Two popular packages are WinGate, a popular proxy system, and SyGate from SyberGen for network address translation. Others are available.

As an alternative, some xDSL modems, which are in fact sophisticated IP routers, can do DHCP, NAT, and firewall filtering. The Cisco 675 RADSL modem and router can be configured for DHCP, NAT, and IP filtering, and is used by many TELCOs such as US West for xDSL service. The Cisco 675 software must be at version 2.0 or later to support NAT.

Dial-up modems

Don't throw out your old dial-up POTS modem and delete your serial PPP configuration just yet. Many companies have private networks that can't be reached over the Internet, and a PPP dial-up connection can still be useful until they offer xDSL access for telecommuters. You can use your POTS modem on the same phone line as your xDSL modem.

The problem you'll encounter when simultaneously running PPP and an xDSL or cable modem is the default route will cause all your IP traffic to go out over your xDSL or cable modem. The trick is to modify your routing tables to indicate that the route to the private network is over the PPP interface and not over the default route to the Internet.

Your default route will typically be through your xDSL or cable modem, as was set with the /etc/defaultrouter file. You'll need to manually add a route to your private network and add the name server for that domain to your resolv.conf or the hosts you wish to reach on the PPP network to your /etc/hosts file. The command `route add -net 192.168.121.0 192.168.121.199` will route all traffic to the 192.168.121 network through your peer PPP host 192.168.121.199. The command `ifconfig -a` can be used to obtain the IP addresses of your PPP configuration.

Routing and security

Many xDSLs and cable modems operate in bridging mode, and the ISP creates separate logical subnets for each modem. This creates the atypical situation where it's impossible to connect to other machines within your own IP subnet, as they're behind a gateway and need to be routed, while they're expected by default to be directly reachable from the interface.

If the gateway doesn't route or proxy ARP to these machines, you won't be able to access machines within your own IP subnet but will on other xDSL modems. You must add routes to these other machines and they must similarly add routes to reach you. This problem typically impacts only gamers and others sharing information on the same ISP network but at different locations.

A significant security issue, beyond that of any full-time Internet connection and static IP address, is that some xDSL and cable modem configurations bridge all subnet traffic to all modems on the subnet, allowing packet sniffing and access to both unprotected TCP/IP and Windows resources. Make sure to secure both TCP/IP and Windows-based network services or set up a firewall using Ip-filter and a host with two Ethernet interfaces.

Summary

xDSL and cable modems have brought affordable high-speed Internet to the small office, home office, and individual. While Solaris isn't typically an officially supported operating system, not only will Solaris operate with xDSL and cable modem systems, but it can also act as a firewall and network address translator for your local network, allowing all your computers to share a single, secure high-speed connection to the Internet.

Web resources

- xDSL modem newsgroup comp.dcom.xdsl
- Cable modem newsroup comp.dcom.modem.cable
- xDSL Technology www.tuketu.com/dsl/xdsl.htm
- ADSL Forum www.adsl.com
- Universal ADSL Working Group www.uawg.org
- Cable Modem Information www.cablemodeminfo.com
- RFC 1918 ftp://www.arin.net/rfc/rfc1918.txt
- Ip-Filter cheops.anu.edu.au/~avalon/ip-filter.html
- Sygate www.sygate.com
- WinGate www.wingate.com

Introducing SNMP, part 1

by Don Kuenz

The Simple Network Management Protocol (SNMP) enables you to monitor and alter performance characteristics of network devices such as workstations, PCs, servers, routers, hubs, gateways, and printers. In this first article of a two-article series, we'll describe the architecture of SNMP. In the next article, we'll show you how to install and use a free software package, which implements the protocol.

Nowadays *network protocol* inevitably means *Internet protocol (IP)*, because over the past couple of years, most vendors abandoned proprietary network architectures in favor of open Internet standards. The *open* part of these standards makes technical information freely available with only a small effort on your part.

The Internet Society (ISOC) appoints people who create protocols, such as SNMP. The ISOC makes its technical activities known by publishing documents called Requests For Comments (RFC). The ISOC names new comments with the letters *RFC* followed by a four-digit number. For instance, RFC1157 contains a comment entitled "A Simple Network Management Protocol (SNMP)," which discusses the original SNMP protocol. You can obtain RFCs at **ftp://ftp.wustl.edu/doc/rfc**. You can also add your own comments on a proposal.

The current version of SNMP is version 2, otherwise known as SNMPv2. As this article goes to print, ISOC members are busy creating SNMPv3 (version 3).

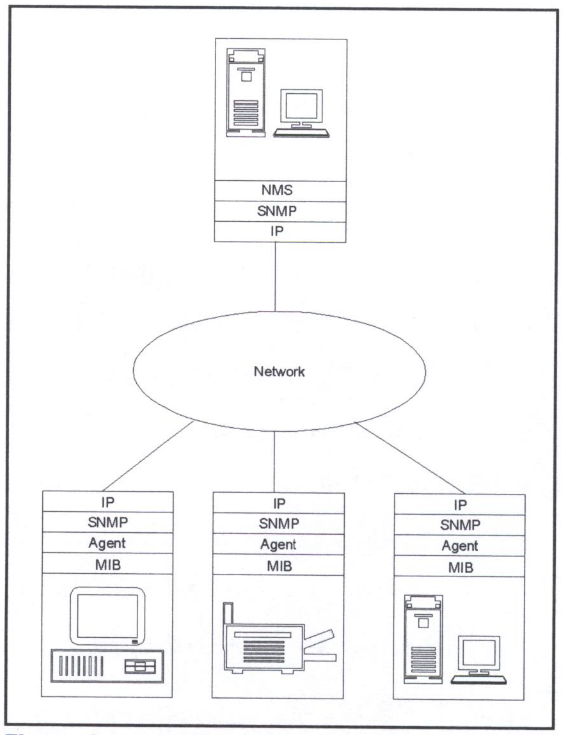

Figure A: *This diagram shows relationships between one Network Management Server and three network devices.*

An overview

This protocol runs at the Open Systems Interconnection (OSI) application layer and allows network devices to read, write, and act upon management data. Management information resides in an object data store and includes everything from the make and architecture of a machine to the average number of data bytes transmitted. Vendors may freely add new object definitions as needed. A data structure named Management Information Base (MIB) defines SNMP objects.

Networked devices fall into two categories: managed devices and Network Management Servers (NMS). An NMS may also function as a managed device. Each managed device contains a local object data store and runs an SNMP application known as an agent. On a Solaris host, the agent is a daemon. The agent's responsibility includes updating the local object data store as events transpire and responding to NMS commands.

In order to keep network traffic to a minimum, NMS hosts perform as much of the network management processing as possible. Most of NMS communications with agents involve reading or writing an object. When an NMS needs data, it reads an object. When an NMS wants to dynamically alter a network parameter, it writes an object.

Reading and writing objects over a network opens the door to a potential security breach, which hackers could exploit to compromise a network. SNMP faces this challenge by logically grouping hosts into network partitions called communities. Host administrators define access privileges for each community allowed access to that host's agent. This effectively limits reads and writes only to known, authorized NMS hosts.

Architecture

Figure A shows a simple SNMP environment, which includes three computers and one printer connected together in a network. The oval in the center represents an intranet, the Internet, or just about any other type of network that supports the IP protocol. Each box represents a device along with the simplified protocol stack used by the device.

A *protocol stack* models the way programmers break networking software code into smaller, more manageable pieces or layers. The lowest layer of a protocol stack transmits bits between hardware devices, while the upper layers take care of things like reliable delivery of packets. We show two types of protocol stacks in **Figure A**. The top box uses a protocol stack suitable for an NMS, while the bottom three boxes use a protocol stack suitable for a managed device.

The computer in the top box runs the NMS software, which interrogates and updates the agents running on the other three networked devices. You can see that the NMS application layer talks to the SNMP protocol layer, which in turn uses IP port 161 to send and receive information from managed devices.

The computers and printer in the bottom boxes run an agent, which makes them managed devices that act upon commands from an NMS host. To simplify the diagram, we actually show an upside-down version of the protocol stack of those three devices. NMS commands arrive as IP packets on default port 161. The SNMP protocol interprets the packets and directs the agent to read and write objects in an MIB data store.

MIB

The heart of network management lies with agent functionality. SNMP describes this functionality using one or more MIBs. You use Abstract Syntax Notation One (ASN.1) to describe objects within an MIB. **Listing A**, on page 8, shows an excerpt of the MIB-II specification (RFC1213 contains a complete definition). SNMPv2, which is the current version of SNMP, uses MIB-II.

Taking a closer look at **Listing A**, we see this excerpt defines a `DisplayString` object named `sysDescr`. The `sysDescr` object displays a description of the managed device. The description shows hardware platform and software operating system. For instance, an agent running on a Solaris Intel host displays:

```
SunOS hephaestus 5.5 Generic i86pc
```

MIB arranges objects into a tree-like, hierarchical structure. RFC 1158 describes a standard MIB for TCP/IP named MIB-II. MIB-II prefixes all groups with `.iso.org.dod.internet.mgmt.mib-2`. Under that prefix, MIB-II branches out into 10 groups: `system`, `interfaces`, `at`, `ip`, `icmp`, `tcp`, `udp`, `egp`, `transmission`, and `snmp`. Each of the 10 groups branch out further, until an object is finally reached. The `::= { system 1 }` code, shown in **Listing A**, denotes that the `sysDescr` object belongs to the `system` group. Tack the standard prefix and the main group on to the front of the `sysDescr` object, add `.0` to the end, and the absolute name becomes `.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0`. Client applications running on an NMS host distinguish an absolute name by looking for a leading period. Many client applications allow the use of a shorter, relative name, such as `system.sysDescr.0`. In the latter case, many client applications notice that the leading period is missing, and will automatically prefix the relative name with `.iso.org.dod.internet.mgmt.mib-2`.

Summary

SNMP allows you to manage network devices on any network that uses IP. Network management includes interrogating managed network devices, as well as dynamically altering their parameters to change performance characteristics.

You perform management functions by running applications on an NMS host. Each managed device contains a local object data store and runs an SNMP application known as an agent. A managed device returns information from its data store during interrogations. It also keeps dynamic parameters in the same store. An NMS host can tune performance by updating dynamic parameters. An MIB defines data store objects to both the NMS and the managed device. SNMP enforces a minimal layer of security by using the concept of a community.

This article covers the basics that you need before you start using SNMP. In our next article, we'll show you how to install and use a freeware software package available from the University of California at Davis.

Listing A: *This is an excerpt from an MIB defined in RFC1213.*

```
RFC1213-MIB DEFINITIONS ::= BEGIN
   IMPORTS
      mgmt, NetworkAddress, IpAddress, Counter, Gauge,
         TimeTicks
         FROM RFC1155-SMI
      OBJECT-TYPE
         FROM RFC-1212;
--  This MIB module uses the extended OBJECT-TYPE macro as
--  defined in [14];
--  MIB-II (same prefix as MIB-I)
mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
-- textual conventions
DisplayString ::=
   OCTET STRING
-- This data type is used to model textual information taken
-- from the NVT ASCII character set.  By convention, objects
-- with this syntax are declared as having
--
--      SIZE (0..255)
PhysAddress ::=
   OCTET STRING
-- This data type is used to model media addresses.  For many
-- types of media, this will be in a binary representation.
-- For example, an ethernet address would be represented as
-- a string of 6 octets.
-- groups in MIB-II
system       OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
at           OBJECT IDENTIFIER ::= { mib-2 3 }
ip           OBJECT IDENTIFIER ::= { mib-2 4 }
icmp         OBJECT IDENTIFIER ::= { mib-2 5 }
tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
udp          OBJECT IDENTIFIER ::= { mib-2 7 }
egp          OBJECT IDENTIFIER ::= { mib-2 8 }
-- historical (some say hysterical)
-- cmot      OBJECT IDENTIFIER ::= { mib-2 9 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp         OBJECT IDENTIFIER ::= { mib-2 11 }
-- the System group
-- Implementation of the System group is mandatory for all
-- systems.  If an agent is not configured to have a value
-- for any of these variables, a string of length 0 is
-- returned.
sysDescr OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
      "A textual description of the entity.  This value
      should include the full name and version
      identification of the system's hardware type,
      software operating-system, and networking
      software.  It is mandatory that this only contain
      printable ASCII characters."
   ::= { system 1 }
sysObjectID OBJECT-TYPE
   SYNTAX  OBJECT IDENTIFIER
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
      "The vendor's authoritative identification of the
      network management subsystem contained in the
      entity.  This value is allocated within the SMI
      enterprises subtree (1.3.6.1.4.1) and provides an
      easy and unambiguous means for determining `what
      kind of box' is being managed.  For example, if
      vendor `Flintstones, Inc.' was assigned the
      subtree 1.3.6.1.4.1.4242, it could assign the
      identifier 1.3.6.1.4.1.4242.1.1 to its `Fred
      Router'."
   ::= { system 2 }
sysUpTime OBJECT-TYPE
   SYNTAX  TimeTicks
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
      "The time (in hundredths of a second) since the
      network management portion of the system was last
      re-initialized."
   ::= { system 3 }
sysContact OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
      "The textual identification of the contact person
      for this managed node, together with information
      on how to contact this person."
   ::= { system 4 }
sysName OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
```

Coming up...

- Help for Y2K
- Configuring Samba 2.0
- MP3 Software for Solaris

Vi macros for Web publishers

by Alvin J. Alexander

The print business being like it is, I wrote this article back when people around the world were practicing the time-honored tradition of making resolutions for improving their lives. Looking at my life as a Solaris administrator and Webmaster, I'm prompted by a similar question: what can I do to work more effectively?

In my daily work life I manage a variety of remotely hosted Solaris-based Web sites. Among other things, this means that I do a lot of work from the Solaris command line using the vi editor. In this environment, one way I can achieve more work in less time is to improve the way I use this standard Solaris editor.

Just as a good carpenter uses good tools and learns the tricks of using each tool, the vi editor is one of my tools, and I've found ways to make it better for my remote-authoring needs. Making vi work the way I want it to work makes me more efficient, which leads to three other good things: higher productivity, happiness, and higher pay.

In this article, I'm going to share with you the macros I've created to become a better HTML developer. If you like these, you can expand the techniques I'm demonstrating to include other programming languages (Java, JavaScript, Perl, etc.) as you see fit.

vi macros for an HTML publisher

When working with existing HTML files, I often modify small sections of existing HTML code, maybe five to ten lines of code at a time. When modifying existing sections of code, I almost always follow these four steps:

1. Copy the original lines of code to a buffer.
2. Keep the lines of code in the file, but place comment tags around them.
3. Paste the buffered lines of code into a new section, usually just below the old code.
4. Begin modifying the pasted section.

I've followed this process so often that I can do it in my sleep. I think of this as a safe way of modifying the code, because if my changes don't work properly, I can easily switch back to the old code. Conversely, when I'm happy with the changes, I remove the old code.

My first vi macro helps me easily perform the comment function (shown in step two) by converting a line of HTML code like this:

```
<tr><td>yada yada yada</td></tr>
```

into a commented-out line of code like this:

```
<!-- <tr><td>yada yada yada</td></tr> -->
```

This macro inserts the `<!--` characters at the beginning of each line and the `-->` characters at the end of each line.

I use this macro very often, and think of it as the comment macro, so I've assigned it to the keystroke pattern `%c`. (Note that all of my HTML macros begin with the percent character [%].)

Here's how I define the comment macro in my `~/.exrc` file:

```
map %c 0i<!-- ^[A -->^[
```

This macro tells the vi editor how to behave each time I hit the `%c` key combination when I'm in command mode. Specifically, vi should do the following:

1. Move to the beginning of the line (the 0 command).
2. Switch to insert mode (the i command).
3. Insert the <!-- characters.
4. Simulate pressing the [Esc] key, which switches the editor back into command mode.
5. Append the --> characters at the end of the line (using the A command).
6. Simulate the [Esc] key again, taking the editor back to command mode.

I use this macro very often, and it's *much* easier than doing all these steps manually.

Of course, if you're going to insert comments into your code like this, you should also be able to remove them easily. I've created an un-comment macro by mapping the `%u` key sequence to remove comments from the current line like this:

```
map %u 05x$xxxx
```

This macro moves to the beginning of the line (using the `0` command), deletes five characters (using the `5x` command), and then goes to the end of the line and deletes four characters. This works fine for my comments, because my `%c` macro does just the opposite, inserting five characters at the beginning of the line and four characters at the end of the line.

Inserting HTML tags with macros

Once I got started and saw the effectiveness of my first HTML macro, I searched for other macros I could create to improve my productivity. Eventually I created a whole array of vi macros to quickly insert HTML tags into my files. These macros cover everything from creating a new HTML template file to creating pre-formatted tables and individual tags like header and paragraph tags. **Table A** lists the HTML macro tags I probably use most often in my daily work.

Here are two quick examples of how this works; typing `%n` while in command mode puts this text into my current file:

```
<HTML>
<HEAD>
   <TITLE></TITLE>
   <META NAME="KEYWORDS" CONTENT="">
   <META NAME="DESCRIPTION" CONTENT="">
</HEAD>

<BODY>
</BODY>
</HTML>
```

This macro also positions my cursor at the end of the first `<TITLE>` tag, so I can start entering my data right away. I think of this as my new file macro.

My `%t` macro (table macro) puts this data into my current data file:

```
<table border=0 cellpadding=0 cellspacing=0
➥width="100%">
<tr>
<td></td>
<td></td>
</tr>
</table>
```

This is a simple table that starts with one row and two columns. It's easy to expand this to other forms by cutting and pasting as necessary.

Other languages

Once you get started, it's easy to create vi macros for any programming language you're interested in. The techniques are the same—only the implementation changes. Probably the worst part about the macros is trying to determine which keystrokes you can use. (Remembering the keystrokes can also be a problem for which I developed a simple cheat-sheet.)

Personally, I use the percent character (%) to start all my HTML macros, the carrot character (^) to start my Java macros, and the ampersand character (&) to start my JavaScript macros. (There's no special rhyme or reason here; they're just keys that are easy to reach, and while I can't always remember a macro command, I can remember that the keys are in alphabetical order: HTML, Java, JavaScript. Use whatever keys you prefer.)

Here's a sample of my `.exrc` file, showing the commands that I described in **Table A**:

```
map #1 :!more ~/.vi_help^M
map %c 0i<!-- ^[A -->^[
map %u 05x$xxxx
map %t o<table border=0 cellpadding=0
➥cellspacing=0 width="100%">^M<tr>^M<td></t
d>^M<td></td>^M</tr>^M</table>^[
map %p 0i<P>^[A</P>^[
map %n :r /html.template^[1Gdd3Gww
map %r o<INPUT TYPE="radio" NAME="
➥" VALUE="">^[
map %f o<FORM METHOD="POST" ACTION=
➥"/cgi-bin/">^M</FORM>^[
```

The first macro in the `.exrc` file maps my [F1] function key to display a Help screen. It accomplishes this by running the `more` command on a file named `.vi_help` in my `HOME` directory. The other commands implement my HTML macros.

While these commands work great, I'm currently working on a consistent set of macros to insert code fragments for the popular Perl `CGI.pm`

Table A: *Common macro tags*

Macro	Behavior
`%f`	Insert my `<FORM>` template
`%n`	Create a new HTML file from my predefined template
`%p`	Insert `<P>` and `</P>` around the current line
`%r`	Insert a radio button
`%t`	Insert a pre-formatted table

module. If you're a Perl/CGI developer, just send me an email at **aja@missiondata.com** and I'll send you my latest macros for this module.

Conclusion

In my business life, I serve as a Webmaster for a variety of remotely hosted Solaris-based Web sites, and in this environment the vi editor is one of my main tools. Just as any good tradesman uses good tools, I'm determined to make vi a powerful tool that works better for me. These macros are one way I make vi a better editor.

I encourage you to look at how you spend your eight hours, or more, of work each day, and determine methods you can use to save the most time, making you and your company more effective. You may find small techniques like this one that save 15 to 20 minutes each day, or perhaps other techniques that automate major tasks through a crontab file, so you'll never need to do them manually again.

PERLing around security

by Paul A. Watters

Larry Wall's *Practical Report and Extraction Language* (PERL) is quickly becoming the industry standard as an enhanced and integrated replacement for several key Solaris utilities, such as *awk* and *sed*. In fact, many of the routine system tasks previously executed through shell scripts are now being undertaken by PERL programs. In particular, PERL scripts are often used for security-related tasks, such as processing password files, because of the language's ability to perform operations efficiently on system databases (such as those created with DBM). In this article, we review some uses for PERL in security-related applications, and address some of the security concerns raised recently regarding the use of an interpreted language for distributed operations in a networked environment.

Why PERL?

PERL is an interpreted programming language designed for UNIX, which has both functional and object-oriented design features, and which has, in many cases, replaced shell-based programming. Its C-like syntax, combined with modern scalar and array data representation and replacing ancient COBOL report writing programs with format templates, has ensured that PERL is the pragmatic administrator's tool of choice.

In traditional shell-based programming, the sequence of events is typically handled by the shell. Although this execution model is modular—such that large applications can be built up from a number of smaller scripts or utilities—every time one of these utilities or scripts changes, the shell script and the external components it relies on may need to be modified.

PERL has syntactic and semantic complexity to rival many popular compiled languages. Thus, it relies less on programs written in external applications that are simply executed in a specified order. This reduces the potential for errors in one external module affecting an entire application. Therefore, it's an ideal language for writing applications to perform with many system administration tasks, especially those involving system databases and security monitoring.

Processing passwords

Let's take a simple example of using PERL to extract some useful information from the system password file. We'll make use of the PERL `split()` operator, which separates database fields and assigns its respective value to scalar or array variables. Let's suppose we wanted to extract a list of encrypted passwords from /etc/passwd (or /etc/shadow, depending on our OS version), which will be passed to a password-guessing program. This task is often performed to warn local users of easily guessable-passwords. Here's an example of all the fields that we'd need to read, split, and then extract the password field from:

```
pwatters:gh5jkjgl:1001:10:Paul A.Watters:/
➥people/pwatters:/bin/tcsh
maya:hty5kfd9:1002:100:Maya F. Watters:/
➥people/maya:/bin/bash
```

Listing A shows a simple PERL program to perform this task. The output from the program is:

```
gh5jkjgl
hty5kfd9
```

Let's walk through the program step by step. First, the location of the password file is specified, and PERL then attempts to open the file for reading, and exits with an error message if the file doesn't exist. A loop is then executed for the number of lines in the password file using the `while()` statement. This loop assigns the values of the password database fields to a set of scalar variables, one of which is `$passwd` (the encrypted password field), which is then printed to standard output with each pass through the loop.

The nice feature of PERL, as opposed to some compiled languages, is that if we wanted to use array rather than scalar variables to store information about passwords, we could simply declare an array like `@passwd`, and assign values read from the password field to indexed locations (for example, `$passwd[$i]`).

Listing A: *Password processing program*

```
$passwdfile = "/etc/passwd";
open (PASSWD, $passwdfile) || die "No password file found!";
while (<PASSWD>)
{
      ($username,$passwd,$userid,$groupid,$fullname,$homedir,
      ➥$shell) = split(/:/, $_);
      print $passwd."\n";
}
close(PASSWD);
```

Listing B: *Modified password processing program*

```
$passwdfile = "/etc/passwd";
open (PASSWD, $passwdfile) || die "No password file found!";
while (<PASSWD>)
{
      ($username,$passwd,$userid,$groupid,$fullname,$homedir,
      ➥$shell) = split(/:/, $_);
      if ($groupid<100)
      {
         print $passwd."\n";
      }
}
close(PASSWD);
```

There's no need to manually allocate and free memory for array variables; PERL will simply use available system memory, and automatically take care of memory management. This means that writing a program to perform a simple task (like retrieving passwords) doesn't require more than a few lines of PERL code to be clean and efficient.

More complex tasks can also be accomplished with PERL. Let's suppose that we only wanted to attempt to guess the passwords of a particular group of users (say with a group lower than 100). We could easily modify the program to extract these passwords. **Listing B** shows the modified program.

In this case, the password field is only printed when the group ID is less than 100. The output from the program is:

```
gh5jkjgl
```

Other possible combinations might include checking whether users are logging in with an appropriate shell, or creating a list of users, full names, and email addresses for an X.500 index.

Security issues

Although PERL can be successfully used for executing security-related tasks, there's always a caveat when using interpreted languages for performing automated tasks in a networked computing environment. PERL is no exception in this respect, although it has largely been the non-UNIX ports of PERL which have experienced difficulties. For example, arbitrary arguments may be passed through the common gateway interface (CGI) to non-UNIX based PERL interpreters located in the cgi-bin directory, with potentially disastrous consequences.

One solution to ensure secure usage of PERL is to force the interpreter to treat any data external to the program as potentially tainted, hence taint mode operation. This means that when PERL is executed with the -T option, only external data and operator calls that are explicitly authorized by the programmer are allowed. For example, attempting to access the sendmail program through CGI to mail the password database to a rogue user will fail if taint mode is enabled and such `system()` calls are forbidden. Using PERL in a secure way will enhance its potentially great productivity value as a tool for writing security-related applications.

PING THE SOLARIS DUDE: SOLARIS Q & A

by Robert Owen Thomas

sar(1) not working

OK, if sar(1) is such a great tool, why do I get "sar: can't open /var/adm/sa/sa25" when I type the sar command?

anonymous
via email

The `sar(1)` command is one of the most powerful system monitoring tools provided with the Solaris operating system. Using `sar(1)`, a Solaris administrator can uncover the performance bottlenecks hidden in the system. Further, `sar(1)` data, tracked over the long-term, can even be used to justify system upgrades from disks, memory, to CPUs. Yet, out of the box, `sar(1)` isn't enabled.

By default, the `sar(1)` collection mechanisms are not enabled. They can be easily enabled, however, in the user sys crontab file. This file, named /usr/spool/cron/crontabs/sys, contains everything you need to get `sar(1)` up and running.

To start the collection of `sar(1)` data, follow these easy steps:

```
su to or login as root.
su - sys
Crontab -e
```

Uncomment the last three lines of the file, and save it.

That's it! Now the sar processes are gathering data. For example, you might want to view the memory utilization over time. You could then issue the command:

```
: pudge; sar -r

SunOS pudge 5.6 Generic sun4u 01/25/99
00:00:00 freemem  freeswap
09:00:00      1251         199192
09:20:00      1300         205152
09:40:00      1588         206937
```

And so on. You can also use the `sar(1)` command to measure context switching, IO statistics, and CPU utilization. Turn to the `sar(1)` man page for further uses of this very useful command.

Limiting /tmp

I use tmpfs for my /tmp directory, but all the files there really eat up a lot of space. How can I limit the amount of swap space /tmp will use?

anonymous
via email

One of the great benefits of the Solaris operating system is the use of swap space for the /tmp file system. While this feature makes access to files in /tmp much faster, it can also lead to constraints on space as either swap, files in /tmp, or both access bytes from the swap area.

One way to avoid resource contention between swap allocation and file allocation in /tmp is to limit the amount of space allocated to /tmp. This can be done in /etc/vfstab with the mount-time option size. You'll need to decide how much space you wish to allocate to /tmp in kilobytes, megabytes, or bytes. Then, edit /etc/vfstab and make the following addition to field seven of the /tmp mount line:

```
swap  -   /tmp   tmpfs   -   yes  size=150m
```

Here we have designated that /tmp will receive no more than 150 megabytes of space for file and directory allocation. Of course, keep in mind that swap space is dynamic, and changes with the demands on memory resources. This will, however, ensure that files and directories in /tmp will never consume more than 150 MB of swap space.

Also, keep in mind that swap space is volatile storage. All the files and directories in swap-based /tmp will be lost at the next reboot.

Required dynamic libraries

A developer recently claimed that several libraries were required for a third-party application he was developing. How can I determine which shared libraries an application requires?

anonymous
via email

The move away from statically linked libraries to shared libraries was certainly a wise move for most binaries. Although certain cases can be made for statically linked libraries, shared libraries benefit the system and the applications by decreasing the overall use of memory and increasing application performance.

To determine which shared libraries an application will require, Solaris provides the `ldd(1)` command. Using `ldd(1)` will provide you a complete list of the required shared libraries, like so:

```
: pudge; ldd /bin/ls
        libc.so.1 =>     /usr/lib/libc.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
```

From the output of `ldd(1)`, it's clear that `libc` and `libdl` will be required for the `ls(1)` command. The `ldd(1)` command provides many other options, so be sure to peruse the `ldd(1)` man page.

If you can't find an answer for that thorny Solaris or Solaris-based application question, drop the Solaris Dude a note today! Ping the Solaris Dude at robt@cymru.com.

How long did it take to run?

Quick tip

by Alvin J. Alexander

Sometimes when you're working with programs that take a while to run, timing the programs can help you understand where the performance problems are. In cases like this, the Solaris `timex` command can help you time the speed of your programs.

As an example, I recently wrote a PERL program to analyze the access_log file from Apache Web servers. Once I got the program working properly, I set off to optimize the code. One of the first steps in optimizing the code is understanding how long it takes originally.

I used the `timex` command to provide a benchmark like this:

```
timex readlog
```

When my program finished running, the output was:

```
real    0m12.77s
user    0m11.40s
sys     0m0.76s
```

This tells me that the application took 12.77 seconds of CPU time to run. Out of that time, 11.40 seconds were spent in user mode, and 0.76 seconds were spent in system mode. Using other `timex` options, you can see other information, such as blocks read/written, main memory size, CPU factor, etc.

After making some changes to the code, I ran this again (and again), until I got the total time down to a point that I thought was reasonable. Using the `timex` command alone, I was able to make changes to my code and easily see the results of my efforts.

Enhance your login prompt: keep users informed with etc/issue

by Alvin J. Alexander

If you have a large percentage of users that log on to your Solaris system via serial terminals or telnet sessions, you can easily customize the appearance of the login prompt they see when they first log in. Personally, I use this feature to make my customers' systems more friendly and informative.

Friendliness may seem like a vague computing concept, but users like it. It also helps build brand identity, which is a valuable asset whether you're a roving consultant or a system administrator. (Have you ever noticed the full-screen advertisement that's displayed every time you turn on a Microsoft Windows PC?)

Looking at this issue from another perspective, controlling the message displayed at the login prompt is a very important issue if you run a government facility. This is a very appropriate place to display warnings and other messages to users *before* they try to log on to your system.

Changing the login prompt

Changing the appearance of the login prompt is easy. If you log on to a Solaris 2.x system via a telnet session, your default login prompt looks like this:

```
UNIX(r) System V Release 4.0 (visitor)

login:
```

(The name of *your* network node(s) will appear in the parentheses.) By simply modifying a file named `/etc/issue`, you can make your prompt look more like **Listing A**.

Listing A: *Improved loging prompt*

```
UNIX(r) System V Release 4.0 (visitor)

#----------------------------------------------------------#
#  Welcome to "The Visitor", we hope you enjoy your stay.  #
#----------------------------------------------------------#
#  Today is Thursday, April 15, 1999.                      #
#  (Only 259 days 'til the millennium bug kicks in.)       #
#----------------------------------------------------------#
#  There are currently 23 users logged in.                 #
#                                                          #
#  P.S. - Remember, tonight's party is at 6:30 p.m.!       #
#----------------------------------------------------------#

login:
```

About our contributors

Alvin J. Alexander first began his career as an aerospace engineer. He's now the president of Mission Data Corporation, an employee-owned computer consulting firm in Louisville, Kentucky. You can reach him online at aja@missiondata.com.

Richard Auletta is currently helping construct the silicon chips that pave the information superhighway. He can be reached at rauletta@orci.com.

Don Kuenz programs for Computing Resources Company in Casper, WY. You can find out more at gtcs.com/crc.

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.

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.

Inside Solaris

Tips & 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 Editor	Jill Suhm
Copy Editors	Rachel Krayer
	Christy Flanders
	Taryn Chase
Contributing Editors	Alvin Alexander
	Richard Auletta
	Don Kuenz
	Robert Owen Thomas
	Paul Watters
Print Designer, Cover and Content Design	Lance Teitsworth
General Manager	Jerry Weissberg
Editor-in-Chief	Joan Hill
Editorial Director	Michael 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@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.

Postmaster

Periodicals postage paid in Louisville, KY.

Postmaster: Send address changes to

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

Price

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

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

Back Issues

To order back issues, 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.

Creating /etc/issue

By default, the `/etc/issue` file doesn't exist on Solaris systems—which makes it hard to know about this feature—but as the root user, you can easily create it with vi or another text editor. Just enter a command like this at your prompt:

```
vi /etc/issue
```

and begin entering the message you'd like your users to see at the login prompt.

If you prefer, you can take this approach a little further and do what I've done. As you can see from that example, the login message is dynamic and displays a new message each day. To enable this effect, I've created a shell program and crontab entry that modifies the `/etc/issue` file automatically.

All you need to do is to create a program that generates the output you want your login users to see, and overwrite the `/etc/issue` file with that output. Here's a simple Bourne shell program to help you get started:

```
#!/bin/sh
TARGET=/etc/issue

echo "`date`" > $TARGET
   echo "Welcome to `uname -n`" >> $TARGET
   echo "" >> $TARGET
```

Just give this program a name, like `/usr/local/bin/create-issue-file`, make it executable, and then put it in root's crontab file. In the case of this program, make sure it's run frequently so the date information is up to date. Of course, you can generate the `/etc/issue` file from all different types of information sources, and a Bourne shell program is just one way to get started.

Testing your message

Once your message has been entered into the `/etc/issue` file, use the telnet command to connect back into your local server to see your message:

```
telnet localhost
```

If everything works as expected, your new login message will be displayed:

```
UNIX(r) System V Release 4.0 (visitor)

Thu Dec 17 10:13:28 EST 1998
Welcome to visitor

login:
```

I recommend this technique if you'd like to make your site more interesting and/or informative for your end-users. Depending on your user base, they may appreciate your extra effort.

lslk

TOOL OF THE MONTH

by Robert Owen Thomas

The tool of the month is *lslk*, or the lock lister utility. You may already be familiar with the lsof tool, which lists open files per process. The lslk utility will list all locks held by processes.

With the lslk utility, a Solaris administrator can easily determine which processes are holding locks on files throughout the filesystem. This can be crucial when certain applications (such as sendmail(1M)) fail due to lock acquisition issues. The lslk utility is a perfect complement to the lsof utility, and it's certainly a tool every Solaris admin should have. You can obtain lslk at **http://ftp.unicamp.br/pub/unix-tools/lslk/**.

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 him a line at **robt@cymru.com**.