ZD JOURNALS

Volume 6, Number 2 • February 2000
www.zdjournals.com/sun
US $9.00

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Benchmarking and performance measurement

by Paul A. Watters

We often make software and hardware purchasing decisions on the basis of promised or advertised performance characteristics. Alternatively, management often asks for commitments to particular performance characteristics in a deliverable software product. The reason for this emphasis is clear: customers want to maximize their value for money. Companies often choose between two competing products if one offers more functionality or better performance.

How do you evaluate the promises made by companies about software performance, and how do you provide realistic advice to management about the characteristics of your own software? In this article, we will look at some quantitative methods to use when establishing software performance characteristics running on the Solaris operating system.

Speed

There are three main performance characteristics: speed, capacity, and reliability. Execution speed is typically the characteristic that you think of when you hear the word performance. **Figure A** shows a typical speed performance chart, comparing unoptimized and optimized Java code.

It's very important for some software packages to minimize the execution time of any one component. For example, a program that attempts to predict movements in stock prices may have a physical limit to its usefulness imposed by the market. The fastest program in this case will probably have the greatest influence.

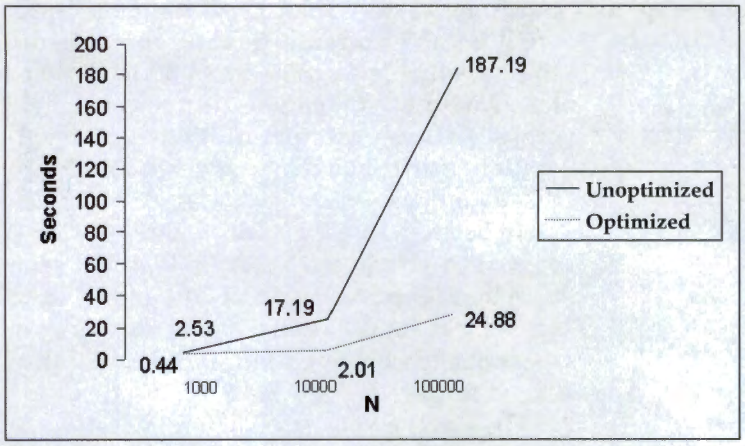


Figure A: *We need this comparison of time to write data elements with unoptimized and optimized Java code.*

Capacity

Capacity is another important characteristic that's often overlooked by benchmarks. As an analogy, let's say we designed two cars. Car A can travel at 110 mph, but only seats two passengers. Car B is limited to 55 mph, but can seat eight passengers. On an open road with no speed limits, Car A would clearly be optimal.

ZIFF-DAVIS a SOFTBANK company

However, given a speed limit of 55 mph, Car A and Car B have equal performance, if only one or two passengers required transportation. But the slower Car B would actually be more useful if three or more passengers required transportation.

In a real-time data processing scenario, an E450 may be constrained by the speed of data arriving through a peripheral device (for example, a 56K modem). In this case, a machine with a similar disk and memory capacity (for example, an Ultra 2) may be just as appropriate on a cost-benefit scale.

Reliability

The third key characteristic is reliability. In the many benchmarks published in computer magazines and trade journals, figures for the performance of particular applications on a certain configuration (for example, a PC running a Web server might serve more hits per second than an old Sparc 20 doing the same job) are common. However, what's rarely mentioned is some kind of reliability estimate, including, for example, mean hardware failure rate figures for individual components. It may be that the older, slower hardware is 10 times less likely to fail at a given moment than the modern hardware.

Code optimization

In this article, we will focus on speed optimization. You can optimize speed by selecting the appropriate programming language and/or runtime environment for your software, as well as at the compiler level. The latter can often produce improvements in speed performance.

Both C compilers and Java runtime environments (using HotSpot technology) attempt to use optimization technology to improve speed. This is achieved by several methods: the overhead in calling loops can be reduced by loop unrolling, for example, which creates a larger binary, but minimizes CPU-intensive iterations.

The GNU C compiler (gcc) has three global levels of optimization that are often used in preference to specifying individual options. Java's new Hotspot technology promises to speed up the execution time of some fairly poor performance characteristics of Java.

One of the most useful Solaris commands for benchmarking is the `time` utility, which measures the elapsed time between the execution of a program and its termination, as well as the User CPU time and System CPU time. This means that by issuing a command like `unix% time ls`, you can actually measure the time it takes for the system to process the command and the time lag between execution and the printing of files in the current directory, in this example. This is the command we used to time the following examples.

A case study

Imagine that you've just been handed a requirements document for a brand new database product. Input/output will need to be very fast. The product will be developed and deployed on the Solaris platform, taking advantage of fast SCSI and RAID performance using StorEdge products. However, there's a cover letter from Marketing on the requirements document which hints that Java is very popular in the IT world, and that we should have a good reason for not using it on this product.

Rather than resorting to marketing hype or press releases to answer this question, we can perform some simple benchmarks on the read/write performance of Java versus C on a Solaris system. Although Java will certainly be slower than C without HotSpot, we'll be able to better justify our decision to Marketing if we can say, "I/O is 10 times faster with C." Of course, on projects where cross-platform compatibility is a consideration, using Java may outweigh any speed disadvantages.

To begin evaluating performance, you need to write a simple test program that will compare the I/O performance of Java and C. To do so, you can simply open a file, write a series of sequentially numbered records to it, and then close the file.

The programs ctest.c, shown in **Listing A**, and jtest.java, shown in **Listing B**, perform these operations. Two programs containing

Listing A: *ctest.c*

```
#include <stdio.h>

main()
{
int i;
FILE *out;
if((out=fopen("ctest.txt","w")) != NULL)
for (i=0; i<1000; i++)
fprintf(out,"Data element %i\n",i);
fclose(out);
}
```

1,000 sequentially numbered entries generate ctest.txt and jtest.txt in their current form. Although this code is much simpler than the requirements of an actual database system, you can use these simple I/O operations as a much-simplified model of a complete system.

We compared the speed of execution of the two programs when 1,000, 10,000, and 100,000 entries were written to the disk. The ctest.c program wrote 1,000 entries in 0.02 seconds (elapsed), 10,000 entries in 0.05 seconds, and 100,000 entries in 0.57 seconds. This data represents a fairly linear increase in the time required to complete the task as a function of the number of items to be written, suggesting that this program would adequately scale up to even larger data-writing requirements.

However, the performance responses for Java were, as expected, much longer. Writing 1,000 items took 2.53 seconds, 10,000 items took 17.19 seconds, and 100,000 items required 187.19 seconds. Again, there was a fairly linear increase in the time required to write items to disk as a function of the number of items. Surprisingly, there was not a significant lag, which could be attributed to the loading of the VM. The processing time for 100,000 items was a simple multiple of the time required for writing 1,000 items. This suggests that the I/O performance of Java would not necessarily improve with the writing of a large number of items, effectively sharing the VM load time overhead.

When revisiting the ctest.c program, we wondered how much of a performance improvement we could gain by using the optimization facilities of the GNU C compiler. We found that this change resulted in no substantial time performance improvement for writing lists of 1,000 or 10,000 items (with times of 0.02 seconds and 0.05 seconds respectively). However, with a time of 0.40 seconds to write 100,000 items, there was a substantial improvement over the standard compiled binary. We can conclude that performance gains through optimization might not be apparent unless you choose an appropriate scale for comparison and testing purposes.

We can realize a similar scale of performance benefit for Java applications by closely examining the features provided by different classes. For example, taking advantage of the disk buffering features provided by the `java.io.PrintWriter` class realizes a significant benefit over unbuffered I/O. In comparison to the slow write times given for 1,000, 10,000, and 100,000 items for jtest.java, these fell to 0.44, 2.01, and

Listing B: *jtest.java*

```
import java.io.*;

public class jtest
{

public static void main (String[] args) throws IOException
   {

DataOutputStream out = new DataOutputStream (new
➥ FileOutputStream("jtest.txt"));
      for (int i=0; i<1000; i++)
          out.writeChars("Data Element " + i + "\n");
      out.close();
   }
}
```

24.88 seconds, respectively, when PrintWriter was used. Only a simple change to the jtest.java code is required. The line

```
PrintWriter out = new PrintWriter(new
➥FileWriter("jtest.txt");
```

replaces the line

```
DataOutputStream out = new
➥DataOutputStream(new
➥FileOutputStream("jtest.txt"));
```

Figure A compares the performance of the optimized versus unoptimized Java code.

Of course, a simple analysis of speed performance should include many samples measured at randomly selected dates and times. This is particularly important on a multi-user, multiprocess machine like the Ultra 2 platform, where we compiled and executed these programs. Mean execution times for a large number of samples are likely to be more meaningful to a marketing department than a single number.

Conclusion

Although we've focussed specifically on time performance and optimization methods in this article, it can be misleading to simply focus on speed at the expense of other considerations. For example, a Java servlet offers an object-based design architecture, data persistence, and guaranteed cross-platform capability compared to a server-side C application, which may execute many times as fast as the servlet. This doesn't mean that all C programs are good, or that all Java programs are bad, or vice versa. The message that we have to communicate to non-technical colleagues is performance analysis based on actual observations for determining the suitability of software for a specific purpose.

For further information about the science and art of performance analysis, check out Richard Grace's book, *The Benchmark Book*, published by Prentice-Hall.

Securing systems with ASET

by Alan Orndorff

Some of the more common tasks performed by you, the system administrator, revolve around security. And there are many tools available that can help you with these tasks, including TCP Wrappers, Satan, and Tripwire. However, the Automated Security Enhancement Tool (ASET), included with version 7 of Solaris, is designed to have the system do some of the more mundane system management tasks, including tightening up the file system and simply reporting that user root has no password, so you don't have to.

You can run ASET on one of three different security levels and it performs seven system checks. Since the routines ASET performs can be disk- or system-intensive, we recommend that you set up ASET to run when the system load is light.

The fundamentals

To begin using ASET, the first thing you need to do is to ensure that the ASET package is installed on your system. To do so, type the following command:

```
pkginfo | grep SUNWast
```

The system will return the following if the package is installed:

```
system      SUNWast
   Automated Security Enhancement Tools
```

If the package is not installed, pkginfo will not return anything.

Security levels

You can set ASET to run at one of three different security levels: low, medium, or high. At the *low* security level, ASET ensures that the security modes of the files are set to those assigned as standard release levels. The utility makes several checks and reports on potential security areas of concern. At this level, ASET makes no changes to your files or system services.

At the *medium* security level, ASET modifies some of the system file permissions and parameters to make the system more secure. ASET reports on security weaknesses and any modifications that it made. At this level, the utility doesn't affect system services.

At the *high* security level, ASET modifies many system files and parameters to minimum access permissions. Most applications should continue to run, but you should test to make sure this is the case. When run at the high security level, ASET also ensures that the system is set up properly to act as a firewall. However, when you set ASET to run at the high security level, your system becomes not only more secure, but also more restrictive in what users can do. System security takes precedence over system usability.

Even though you can configure ASET to run at three security levels, it won't change the permissions of a file to make it less secure unless you tell ASET to run at low security level. For example, if you have a file with the permissions set to r--r--r-- at the medium level of security, ASET thinks that filename should have permissions of rw-rw-rw-. Therefore, it will change the permissions to rw-rw-rw-, which is less secure than r--r--r--.

Directories

You'll find ASET under the /usr/aset directory structure. All configuration files for the utility are also under this directory. However, you can configure ASET to use files that reside elsewhere on the network.

The seven tasks

ASET performs seven tasks. We'll discuss each of the following tasks in detail:

- System Files Permissions Verification
- System Files Checks
- User/Group Checks
- System Configuration Files Check
- Environment Check
- eeprom Check
- Firewall Setup

System Files Permissions Verification

This task sets the permissions on system files to the security level you designate. This check is also done when you first install the system to set the baseline for the system. At the low security setting, file permissions are set for an open information sharing system. At the medium security setting, permissions are tightened and should be adequate for most environments. At the high security setting, permissions are severely restricted. Any modifications that are made by this task are recorded in the file tune.rpt.

System Files Checks

This task examines system files and compares them to a master list. There's one master list for each level of security. You can modify the master list for each security level to your individual preference. The following items are checked for in each file in the master list for that security level:

- Owner and group
- Permission bits
- Size and checksum
- Number of links
- Last modification time

Any discrepancies are reported in the file cklist.rpt.

User/Group Checks

This task checks the consistency and integrity of user accounts and groups defined in either the local files, NIS, or NIS + files. NIS or NIS + password file problems are reported but are *not* corrected. This check looks for the following:

- Duplicate name or IDs
- Entries in incorrect format
- Accounts without a password
- Invalid login directories
- The nobody account
- Null group password
- A plus sign (+) in the /etc/passwd file on a NIS or NIS + server

Any discrepancies are reported in the file usrgrp.rpt.

System Configuration Files Check

This task checks the following files:

- /etc/default/login
- /etc/hosts.equiv
- /etc/inetd.conf
- /etc/aliases
- /var/adm/utmp
- /var/adm/utmpx
- /.rhosts
- /etc/vfstab
- /etc/dfs/dfstab
- /etc/ftpusers

ASET performs various checks and modifications on these files and reports all problems in the file sysconf.rpt.

Environment Check

This task checks to see how the PATH and UMASK values are set for users of the system. It checks .profile, .login, and 0.cshrc files. Any discrepancies are reported in the file env.rpt.

eeprom Check

This task checks the value of the eeprom security parameters to ensure that they are set to the appropriate security levels. ASET doesn't change these values, but reports its recommendations to the file eeprom.rpt.

Firewall Setup

This task ensures that the system can be safely used as a firewall. The firewall task runs at all security levels, but only takes action when ASET is run at the high level of security. If you want to run ASET at the high level of security but don't want your system set up as securely as required by a firewall machine, you can eliminate this task. Any changes made by this task are recorded in the file firewall.rpt.

Configuring ASET

Each aspect of ASET can be customized using various configuration files. These configuration files control security levels and directory coverage. They also determine each task that's executed.

Master files

The following files reside in /usr/aset/masters and help control how ASET runs:

- tune.low
- tune.medium
- tune.high
- uid_aliases

The three tune files correspond to each of the ASET security levels and contain the permissions and other attributes of the system files used at each of the three different security levels. You can modify these files to your liking if you want to add, remove, or change the permissions of the files.

The uid_aliases file contains a list of multiple user accounts sharing the same ID. Normally ASET would flag this as an area of concern. You can bypass this warning by placing those accounts into this file.

Checklist files

The system files check task checks attributes of files in selected system directories. You define which directories to check by using these checklist path environment variables:

- `CKLISTPATH_LOW`
- `CKLISTPATH_MED`
- `CKLISTPATH_HIGH`

The `CKLISTPATH_LOW` variable defines the directories to be checked at the low security level. The `CKLISTPATH_MED` and `CKLISTPATH_HIGH` environment variables function similarly for the medium and high security levels.

The directory list defined by a variable at a lower security level should be a subset of the directory list defined at the next higher level. For example, you should include all directories specified for `CKLISTPATH_LOW` in `CKLISTPATH_MED`, and all the directories specified for `CKLISTPATH_MED` in `CKLISTPATH_HIGH`.

The checks ASET performs on these directories aren't recursive; ASET only checks those directories explicitly listed in the variable. It doesn't check their subdirectories. You can edit these variable definitions to add or delete directories that you want ASET to check.

Environment file

The file /user/aset/asetenv has two main sections:

- A user-configurable parameters section
- An internal environment variables section

As you probably guessed, the internal environment variables section should not be changed. However, you can do the following in the user-configurable parameters section:

- Choose which tasks to run
- Specify directories for the checklist task
- Schedule ASET execution
- Specify an aliases file
- Extend checks to NIS or NIS + tables

If you don't want to run the firewall task, you could edit this file and remove that task. You can set the directories for each of the checklist tasks here as well.

To start ASET from the command line, you do so with a few switches. One of the switches is `-p`. The `-p` switch tells ASET to run at a specified time. The time that you configure ASET to run is the same as the entries for the `crontab` command.

If you do want to use the standard /usr/aset/masters/uid_aliases file, you can specify another file here. The value of YPCheck determines whether ASET checks NIS or NIS + tables or not. The value is a boolean, either true or false. Remember that ASET will repair normal local password files, but will only report on NIS or NIS + files.

NFS

You can configure ASET to work in an NFS environment. If you're working in a large environment, you can set up several master file areas on an NFS file server and point a machine to use these files instead of the files installed on the local hard disk. This also lets you centrally manage how ASET is configured to run by simply modifying the files on one or more NFS servers.

You can direct all ASET reports from all of your clients or servers to write to an NFS mount point. This allows for a central collection point for all log files.

Environment variables

You can set the environment variables in Table A for use with ASET. Please consult the appropriate man page for the shell you're using if you need help setting variables.

Format of the tune files

Each tune file conforms to the following format:

```
pathname mode owner group type
```

Pathname can be a filename or a wildcard character. The mode of the file is a five-digit octal number that represents the mode of the file(s). The owner is the owner of the file(s), not the owner ID. The group is the group of the file(s), not the group ID. The type can be symlink (for a symbolic link), or directory or file if not a symbolic link.

For the fields owner, group, and type, you can use a question mark (?) instead of filling in a parameter. A question mark tells ASET to leave the setting as is. If you modify the entry in such a way that a file has two entries, the most restrictive setting takes precedence. If you modify the entry in such a way that a file has two owners or groups, the later entry takes precedence.

Format of uid_aliases file

The uid_aliases file conforms to the following format:

```
uid=alias1=alias2=alias3
```

An example could be:

```
0=root=admin=sysadmin
```

Running ASET

You can run ASET interactively or through a schedule. To run ASET interactively, simply issue the command

```
/usr/aset/aset -l level -d pathname
```

where 1 is the security level you wish to run. If omitted, the default of low is used. Pathname is the working directory for ASET. If pathname is omitted, the default of /usr/aset is used.

To run ASET through a schedule, please see the PERIODIC_SCHEDULE information in Table A. To verify that ASET is correctly set up to run periodically, check your crontab entry:

```
crontab -e root
```

Table A: *Environment variables*

Name	Description
ASETDIR pathname	The ASET Working directory.
ASETSECLEVEL level	The security level you want ASET to run at: low, medium, or high.
ASETDIR and ASETSECLEVEL	This cannot be set in the /usr/aset/asetenv file.
PERIODIC_SCHEDULE	Follows the same format as crontab. For example, PERIODIC_SCHEDULE = "5 0 * * 6".
TASKS	Tells ASET which tasks to run. The seven tasks are abbreviated, as shown in Table B on page 8. To set the task's environment variable, enclose the tasks in quotes: `TASKS = "tune usrgrp"; export TASKS`
UID_ALIASES	Specifies which file to use for uid_aliases. For example, UID_ALIASES=/path/uid_aliases.
YPCHECK	If true, reports on problems with NIS or NIS + tables. If false, only checks the local files.
CKLISTPATH_level	Specifies the directories to be checked by each of the related ASET security levels. The path is similar to the shell $PATH in that each directory is separated by a colon.

To stop running ASET periodically, edit your crontab entry and remove the entry for ASET. After removing ASET from the crontab, run crontab –e root one more time to ensure that the entry has been removed. If you run ASET with the `-n` switch, after ASET completes it sends an email message to the user you designated.

Execution log

Whether you run ASET interactively or scheduled, it generates an execution log to standard output. The execution log states the time that ASET started to run, any error messages, and the tasks that were started for this run of the utility. ASET tasks finish asynchronously. If you want to view the output of ASET, you need to wait until all tasks finish.

Taskstat

To see which tasks have finished and which are still running, use the `/usr/aset/util/taskstat` command. If you aren't running ASET from the default working directory of /usr/aset, you'll need to add the working directory to the `taskstat` command. The execution log only notes that a task has started, and not that a task has finished.

To verify that all the data has been collected, you'll need to use the taskstat utility. Another way to verify that a task has finished is to look at its associated results file. When a task has started, its result file will have a banner stating that the task has started. Another banner, at the end of the report, states that the task has finished.

Reports

As with all things defined by management, it isn't over until it's been documented. All reports generated by ASET are stored in the /usr/aset/reports directory. If you go through the listing of the seven tasks of ASET, you'll find the name of the report associated with that task.

The report utility has two nice features. The first is that all log files are stored under a directory that's named after the date and time that ASET was run. This makes it easy to archive past runs and compare them to the current run. The other nice feature is that the latest directory is a symbolic link to the directory that ASET last wrote a report file to. To see the last run of ASET, simply cd to /usr/aset/reports/latest. A sample directory structure may look like the following:

- 0202_1300
- 0204_1500
- latest

This would indicate that ASET has been run twice on this system: once on February 2nd at 1:00 P.M. and again on February 4th at 3:00 P.M. The latest directory would then be symbolically linked to the February 4th entry.

As we mentioned, you can set ASET to centrally store all report files. To do so, on the NFS server under /usr/aset/, create a reports directory:

```
cd /usr/aset
mkdir all_reports
cd all_reports
```

Under this directory, create one directory for each client that you want to centrally collect the reports for. If you have clients named Bugs and Daffy, you would create these two directories now:

```
mkdir bugs_rpt
mkdir daffy_rpt
```

The directory listing ls would look as follows:

```
/usr/aset/all_reports/bugs_rpt
/usr/aset/all_reports/daffy_rpt
```

Now, add these directories to your /etc/dfs/dfstab file. These exports should have the read/write option set:

```
share -F nfs -o rw=bugs
   /usr/aset/all_reports/bugs_rpt
```

Table B: *Tasks that ASET runs*

Task	Description
tune	System Files Permissions Verification
cklist	System Files Checks
usrgrp	User/Group Checks
sysconfig	System Configuration Files Check
env	Environment Check
eeprom	eeprom Check
firewall	Firewall Setup

```
share -F nfs -o rw=daffy
   /usr/aset/all_reports/daffy_rpt
```

Then, issue the `shareall` command. On your client machines, mount these directories to /usr/aset/masters/reports:

```
mount nfsserver:/usr/aset/bugs_rpt
   /usr/aset/masters/reports

mount nfsserver:/usr/aset/daffy_rpt
   /usr/aset/masters/reports
```

If you want this to happen automatically, place the mount commands into the /etc/vfstab file on the client machines. The /etc/vfstab entry would look like the following:

```
nfsserver:/usr/aset/all_reports/bugs.rpt /
➥usr/aset/reports nfs - yes hard

nfsserver:/usr/aset/all_reports/daffy.rpt /
➥usr/aset/reports nfs - yes hard
```

Restoring files

Sometimes things go wrong and you need a quick way to undo a system modification. If you should happen to modify the tune.* files to the point that your system is rendered nearly unusable, you can run `aset restore` to undo the changes made by the last run of ASET. Running `aset restore` also deactivates any scheduled runs of ASET that you have set up in crontab. After you have reconfigured your master files so your system is usable again, you'll need to setup your crontab entries again.

Troubleshooting URL

If you have any error messages that you're having trouble deciphering, point your browser to

docs.sun.com:80/ab2/coll.47.8/SYSADV2/@Ab2PageView/40492?

This Web site contains a listing of ASET error messages and corrective actions.

Conclusion

ASET can be a useful tool to help you automate the security of your system. ASET provides an environment than you can custom tailor to fit your needs. The hardest task will be ensuring that the tune.* files are set up in such a way that leaves your systems secure and usable by the users. Check the log files frequently and pay heed to the information contained in them. The entries are there to help you keep your machines running. If ASET doesn't meet all your security needs, consider using it along with another product to ensure that your systems are safe and being used in the manner that you intend.

/etc/system

by Edgar Danielyan

What's /etc/system? Simply put, /etc/system is the kernel's configuration file. It can modify Solaris installation defaults and set options both in the kernel itself and in various drivers and modules. As the majority of system functions performed by the kernel are configured automatically, it's unlikely that you'll need to edit /etc/system, but if you do, make sure you have a backup copy. It is read once at the system's startup time, so you'll need to reboot after making any changes to it. In this article, we'll explain the format of /etc/system and list basic variables.

Syntax and format of /etc/system

/etc/system is a text file, with lines a maximum of 80 characters long. Comments must begin with an asterisk (*) and must end with CR. By default, commands aren't case sensitive. Table A, on page 10, shows a brief description of available commands. Table B, also on page 10, shows the valid values *for namespace*. Following are some examples using these commands:

```
set maxusers=1000
rootfs: hsfs
set yourdriver:workornot=no
```

Setting kernel parameters

There are a few parameters you may want or need to modify. Table C displays UFS parameters, Table D shows STREAMS parameters, and Table E lists miscellaneous parameters. In addition to /etc/system, you can set certain kernel parameters by hand at the boot prompt.

Following are the options that can be given at boot time:

- `-a`—Instructs the kernel to ask for each and every configuration directive. For example, to skip a damaged or incorrect /etc/system, enter /dev/null when the kernel asks for the configuration file.
- `-s`—Boots to single-user mode.
- `-v`—Verbose boot. That is, all kernel messages are sent to the screen.
- `-r`—Reconfigures the kernel. That is, probe all known hardware and create corresponding files in /dev and /devices for the found devices. The same effect may be reached by using touch /reconfigure.
- `-f`—Applicable only to systems supporting AutoClient feature. Instructs the system to flush client's cache.

This ends our tour of /etc/system. For a more detailed and in-depth description, see "Tuning Kernel Parameters" in the *Solaris System Administration Guide, Volume II*. And don't forget to back up before you make any modifications!

Table A: */etc/system commands*

Command	Description
`exclude: namespace/modulename`	Excludes the named module (that is, don't load it).
`forceload: namespace/modulename`	Forces the loading of the named module at the kernel initialization time. (By default, modules are loaded as and when they are required.)
`rootdev: devicename`	Sets the root device to devicename. (The default rootdev is set by the boot program.)
`rootfs: rootfilesystemtype`	Sets the root file type to rootfilesystemtype.
`moddir: module1, module2, modulen`	Sets the module search path.
`set [<module>:]<symbol>=value`	Sets the variable symbol in module to value.

Table B: *Possible namespace values*

Parameter	Description
drv	Device drivers
exec	Execution format modules
fs	File system modules
sched	System scheduler
strmod	STREAMS modules
sys	System calls modules
misc	Other modules

Table C: *UFS parameters*

Parameter	Description
`ufs_ninode`	Maximum size of the inode table (the default is max_nprocs + 16 + maxusers + 64)
`ncsize`	Number of the directory name lookup cache (default is max_nprocs + 16 + maxusers + 64)

Table D: *STREAMS parameters*

Parameter	Description
`nstrpush`	Number of maximum pushes allowed
`strmsgsz`	Maximum STREAMS message size (0 indicates no limit)
`strctlsz`	Maximum size of the STREAMS control part of the message
`strthresh`	Maximum size of the dynamic memory STREAMS subsystem can use, in bytes (0 indicates no limit)

Table E: *Miscellaneous parameters*

Parameter	Description
`Npty pttys`	Number of SunOS 4.0/4.1 configured (default is 48)
`pt_cnt`	Number of SunOS 5.7 pttys configured (default is 48)

Defining an Acceptable Use Policy

by Edgar Danielyan

Trouble comes without warning. While there are steps you can take to minimize the possibility of such problems as spam, unauthorized access to systems, unauthorized modification of data, and other unpleasant happenings in the life of a system administrator, it is hardly possible to avoid them altogether. In this article, we will try to define the range of problems you may face and give some recommendations for defining an Acceptable Use Policy (AUP) document for your organization, which will help you deal with these problems when they arise.

What's an AUP?

An Acceptable Use Policy is a legal document that defines what your network users can and can't do. An ideal AUP is the one defined and formulated by you (the system or network administrator), your organization's CEO, and your legal counsel. Please also note that the AUP isn't a replacement for a service agreement or contract, but a very important safety measure that both you and your organization need to take.

A good AUP document can save you thousands of dollars and much time, provided it's well written and correctly used. Consider mentioning in the service agreement that the current AUP forms an integral part of the said service agreement and thus is legally binding.

An AUP must be a public document—that is, everyone must be able to get a copy of it at no charge. The best way to publish your AUP is to post it on your Web site and have it sent to your network users along with their network access information. Last, but not least, an AUP must be explicitly dated. In addition, you may want to use a version number to make your life easier, because an AUP is a living document and you probably will make changes to it in the future.

Identifying the parties

There are several parties to the AUP, each having their respective rights and duties defined by the law and your service agreement. Without going into the depths of legal definitions, let's name them. The first is your organization, which owns and/or operates the system or network in question. Then there are network users, both inside your organization (that is, employees, officers, and agents of your organization) and outside users (clients and partners, for example).

It must be noted that you can define rights and duties of these parties only with their consent—that is, you can't (and should not) impose your policy on entities not connected with your organization. This is especially important for networks and systems connected to the Internet and ISP networks, as their users may be hundreds of thousands of miles away in another country and under another jurisdiction.

Defining the rights and duties

A good Acceptable Use Policy must protect you, yet not be too restrictive. Depending on the nature of your organization and services provided by your system, as well as the desired level of security, you'll have to define your rights and duties as well as those of your users. Needless to say, the AUPs of an ISP, a research company, and a government agency may be so different that the only common thing would be the name of the document. However, in all these circumstances, you'll have to answer these general questions in order to define and formulate a good AUP:

- What kinds of service is your organization providing, to whom, in what environment, and how? Do you provide only corporate email services, or complete networking solutions? Are your network's users only inside your organization, or are there entities outside your organization who access your network? If so, what services are they allowed to utilize? When, and how much? Are your network users allowed to use the services provided for their personal communication needs? If yes, how much? If no, what are the penalties?
- What is the procedure for dealing with violations of the AUP? Do you want and need an internal arbitration procedure, or you will resort directly to legal recourse? (When answering these questions, you must seek the advice of your legal counsel!)

- Do you guarantee the quality and quantity of your services? To what extent? What if you fail to provide the guaranteed service? This may be quite an important consideration for, say, a hospital's or a bank's network.

When formulating your AUP, don't forget to harmonize it with your service agreement and other binding promises, such as employment contracts, etc. Also keep in mind that some states don't allow exclusion of certain guarantees with regard to the services provided.

To conclude, the more time and effort you spend on defining, formulating, and refining your Acceptable Use Policy, the less time and money you'll spend on solving the problems when they arise!

Run Linux applications on your Solaris with lxrun

by Clayton E. Crooks II

Have you ever wanted to run a Linux application or game on your Intel-based Solaris machine? It's now possible using a relatively simple and free emulator. Lxrun is an open-source emulator that enables you to run Linux applications unmodified on an Intel platform running the Solaris 7 operating system. In this article, will discuss the background of lxrun, how to set up your Solaris system to run lxrun, and how to download WordPerfect 8 for your use.

Background

The writer's of lxrun original motive was to run Netscape Navigator on SCO OpenServer. When Netscape ported Navigator to SCO platforms, development was halted. Later, work was resumed in an attempt to run Adobe Acrobat Reader on OpenServer. The body of system call mappings in lxrun grew gradually as users modified it to work with more and more applications. By this process, lxrun evolved into a sound emulator.

Solaris engineers have made improvements to lxrun and returned these modifications to the development effort maintained by Steven H. Ginzburg. These enhancements are part of the current release.

Who does lxrun benefit?

Lxrun benefits both users and software developers. Users can expand their current Solaris and Java application portfolio with new Linux applications. This opens an entirely new library of software. Developers see a tremendous advantage in gaining access to multiple platforms without the need to change even a single line of source code. Lxrun is a software layer that sits between Solaris and the Linux Intel binary executable and remaps system calls on the fly allowing them to run without modification on a Solaris machine.

Sun demonstrated this at the 1999 Linux-World Conference and the Linux Expo. Linux applications were running unmodified via lxrun on the Solaris 7 operating environment. These demonstrated applications ranged from word processors and browsers to applications and games. The software included Gnome, WordPerfect 7, WordPerfect 8, Applix, Quake2, GIMP, Netscape Navigator, and Netscape Communicator.

Setting up Solaris to run lxrun

You need to perform several steps to set up a Solaris system to use lxrun. Make sure to read the instructions that you download (see "Web sites of interest" at the end of the article) with the emulator for any last-minute changes or revisions.

First, set up a Linux distribution for the Solaris operating environment to access. You can do this any number of ways, such as from another disk through the Solaris ext2fs file system, from a copy, or directly via NFS from a Linux machine. By default, lxrun will look for this distribution in /usr/local/linux, but you can set the environment variable `LINUX_ROOT` to wherever you put it.

Second, untar the lxrun tar file and either build it from source (preferred) or grab the prebuilt lxrun binary. The Makefile is set up to build for Solaris; whereas the final source from the official Web site will probably use `configure`. Place the binary wherever you like.

Finally, The PATHMAP file tells lxrun how to remap file accesses when the Linux binary tries to open files. For example, if the Linux binary tries to open /etc/passwd, it probably wants to see the Linux password instead of the Solaris /etc/passwd. Thus, lxrun translates a request to open /etc/passwd into a request to open $LINUX_ROOT/etc/passwd. The PATHMAP file controls what's remapped and how. We remap almost everything. By default, it looks for /usr/local/lxrun/PATHMAP, but you can set the environment variable `LINUX_PATHMAP` to the actual lxrun location, such as /home/yourname/lxrun/PATHMAP.

Now you should be able to run lxrun, such as:

```
lxrun /usr/local/linux/usr/bin/netscape
```

Or, if you want to be in a completely Linux world, start the following shell:

```
lxrun
/usr/local/linux/bin/csh
```

One final note: To enable sound, you'll probably need sound drivers from 4Front Technologies (**www.4front-tech.com**). They make Open Sound System drivers, which work on a variety of platforms, including the Solaris operating environment. 4Front Technologies also provide the API that Quake2 uses for sound.

Downloading and installing WordPerfect 8 for Linux

In our testing of this emulator, we downloaded and installed WordPerfect 8 for Linux. We'll discuss the steps we used as well as a few minor troubles we encountered.

Downloading WordPerfect 8

To begin the WordPerfect 8 download, log on to your system as root. Create the directory /home/tmp as a download target directory. Now, cd into /home/tmp.

Now, point your Web browser to **linux.corel.com/linux8/download.htm**. Follow the links to download the evaluation binary (wp8_GUILG00) into /home/tmp. If they aren't already installed on your system, install GNU File Compression Utilities (gzip) and GNU tar (gnu tar) in /usr/local/bin.

Next, modify your `PATH` variable to ensure that gnu tar comes before the native system tar:

```
export PATH="/usr/local/bin:$PATH"
```

Unpack the archive by issuing the following commands:

```
gunzip wp8_GUIL00
tar xvf wp8_GUIL00
```

Unpack the sub-archives by issuing the following shell commands:

```
for i in b_* g_*; do
tar xvf $i
done
```

Installing WordPerfect 8

To install WordPerfect 8, create the appropriate links and installation directories:

```
cd /usr/local
ln -s /usr/lib/linux/usr/local/wp8 wp8
mkdir /usr/lib/linux/usr/local/wp8
```

Now, cd to /home/tmp and run the ./Runme installation script. Answer the prompts as follows:

- `Are files unzipped and extracted ? Y`
- `linux.5 not certified - continue: Y`

After some time, graphical dialog boxes start appearing. Answer each as indicated, clicking OK when done:

- Accept default Lang.
- Accept default License.

- Install Directory: /usr/local/wp8.
- Select Medium install.
- Leave blank Pattern Matching entry.
- Do not update magic file.
- Install English-US language support (default).
- Select printer (system-specific).

Running and troubleshooting WordPerfect 8

To run WordPerfect 8, use the command `/usr/local/wp8/wpbin/xwp`. We recommend that you test the application with simple documents before using in a full production capacity. Try to save and print several documents. Although the known problems that we encountered are few, it wasn't thoroughly tested in a daily use setting.

Known problems

First, on startup, you might see:

```
Cannot create new process. You may have
   exceeded either user or system limits
   for number of active processes or the
   executable may not be found in its proper
   directory.
```

This error is harmless. Simply click OK.

Second, if you exit WordPerfect 8 from the Control screen, it doesn't shut down any other windows or return you to the command line prompt. You can find the process number for xwp and kill it, or use xkill on the remaining WordPerfect 8 windows.

Conclusion

While not 100 percent perfect, lxrun is a must-have for those of you who want to run Linux applications on your Solaris. It's free, easy to use and install, and it works quite well with a diverse set of Linux applications.

Web sites of interest

Lxrun is freely available from the two following Web sites:

- www.sunfreeware.com
- www.ugcs.caltech.edu/~steven/lxrun/

Also visit the Sun-hosted homepage for lxrun located at www.sun.com/software/linux/lxrun/index.html.

ICQ on Solaris

by Clayton E. Crooks II

ICQ (I Seek You), shown in Figure A, is a revolutionary, user-friendly Internet program that tells you who's online at all times, enabling you to contact people at your command. You'll no longer wonder if associates or friends are online. ICQ does the searching for you, alerting you in real time when they sign on. In addition, with ICQ, you can chat, send messages, Web pages, and files, play games, or just hang out with others while surfing the net or using other applications. While you work on other applications, ICQ alerts you when friends and associates sign on, allowing you to work efficiently while having a whole range of Internet functions at your fingertips.

Many functions are available with ICQ, including chat, message, email, URL, and file transfer. ICQ also supports a variety of popular external Internet applications. With ICQ, launching an Internet telephone or video call is a breeze. With a click of a button, you and a friend (or friends) are instantly connected. These events may be executed among multiple users, so you can conduct a conference or just hang out online.

ICQ Java

By far the most popular instant messenger on the planet with thousands of downloads each week, ICQ is also the most feature-packed. The combination of popularity (over 30 million downloads to date) and features make this a must-have for the Solaris community. Unfortunately, there isn't a native Solaris version available for this Windows- and Mac-dominated software. That's where ICQ Java comes in. It's a Java Application of ICQ which, like the full version of ICQ, continually alerts you when your friends and associates sign on to the Internet, and enables you to send messages, chat requests, files, and URLs at will.

Configuring and running ICQ Java

The current version of ICQ Java, beta 0.981a, requires several steps to configure and run. First, you must download the ICQ Java package. Your first stop should be **www.icq.com** for links to download sites and current version information. After acquiring ICQ Java, read any help files that come with the package. The versions are constantly changing, and this might save you some

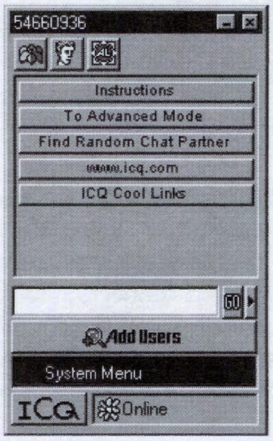
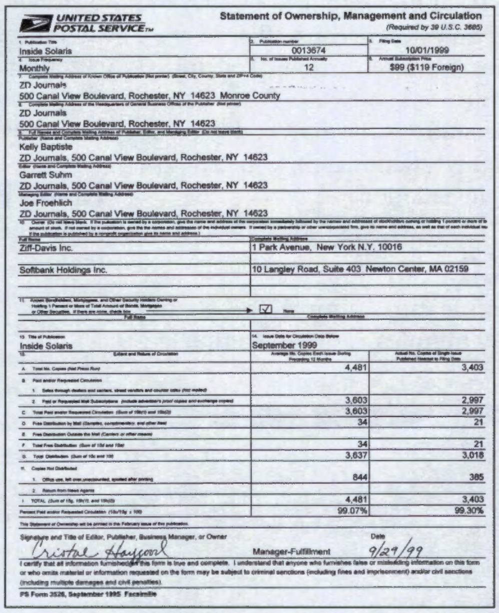


Figure A: *This is how ICQ looks running on your Solaris desktop.*



UNITED STATES POSTAL SERVICE™ — **Statement of Ownership, Management and Circulation** *(Required by 39 U.S.C. 3685)*

1. Publication Title: Inside Solaris	2. Publication number: 0013674	3. Filing Date: 10/01/1999
4. Issue Frequency: Monthly	5. No. of Issues Published Annually: 12	6. Annual Subscription Price: $99 ($119 Foreign)

7. Complete Mailing Address of Known Office of Publication (Not printer) (Street, City, County, State and ZIP+4 Code)
ZD Journals
500 Canal View Boulevard, Rochester, NY 14623 Monroe County

8. Complete Mailing Address of the Headquarters of General Business Office of the Publisher (Not printer)
ZD Journals
500 Canal View Boulevard, Rochester, NY 14623

9. Full Names and Complete Mailing Address of Publisher, Editor, and Managing Editor (Do not leave blank)
Publisher (Name and Complete Mailing Address)
Kelly Baptiste
ZD Journals, 500 Canal View Boulevard, Rochester, NY 14623
Editor (Name and Complete Mailing Address)
Garrett Suhm
ZD Journals, 500 Canal View Boulevard, Rochester, NY 14623
Managing Editor (Name and Complete Mailing Address)
Joe Froehlich
ZD Journals, 500 Canal View Boulevard, Rochester, NY 14623

10. Owner (Do not leave blank. If the publication is owned by a corporation, give the name and address of the corporation immediately followed by the names and addresses of stockholders owning or holding 1 percent or more of the total amount of stock. If not owned by a corporation, give the names and addresses of the individual owners. If owned by a partnership or other unincorporated firm, give its name and address as well as that of each individual owner. If the publication is published by a nonprofit organization give its name and address.)

Full Name	Complete Mailing Address
Ziff-Davis Inc.	1 Park Avenue, New York N.Y. 10016
Softbank Holdings Inc.	10 Langley Road, Suite 403 Newton Center, MA 02159

11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages or Other Securities. If there are none, check box → [✓ None]

Full Name	Complete Mailing Address

13. Title of Publication: Inside Solaris
14. Issue Date for Circulation Data Below: September 1999

15. Extent and Nature of Circulation	Average No. Copies Each Issue During Preceding 12 Months	Actual No. Copies of Single Issue Published Nearest to Filing Date
A. Total No. Copies (Net Press Run)	4,481	3,403
B. Paid and/or Requested Circulation		
1. Sales through dealers and carriers, street vendors and counter sales (not mailed)		
2. Paid or Requested Mail Subscriptions (include advertiser's proof copies and exchange copies)	3,603	2,997
C. Total Paid and/or Requested Circulation (Sum of 15b(1) and 15b(2))	3,603	2,997
D. Free Distribution by Mail (Samples, complimentary, and other free)	34	21
E. Free Distribution Outside the Mail (Carriers or other means)		
F. Total Free Distribution (Sum of 15d and 15e)	34	21
G. Total Distribution (Sum of 15c and 15f)	3,637	3,018
H. Copies Not Distributed		
1. Office use, left over, unaccounted, spoiled after printing	844	385
2. Return from News Agents		
I. TOTAL (Sum of 15g, 15h(1), and 15h(2))	4,481	3,403
Percent Paid and/or Requested Circulation (15c/15g x 100)	99.07%	99.30%

This Statement of Ownership will be printed in the February issue of this publication.

Signature and Title of Editor, Publisher, Business Manager, or Owner: Cristal Haygood Manager-Fulfillment Date: 9/29/99

I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including multiple damages and civil penalties).

PS Form 3526, September 1995 Facsimile

About our contributors

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, TN. He is married with one child. His hobbies include game development, 3-D modeling, and any athletic activity he can find time for.

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

Alan Orndorff has been working with computers since 1990. He is using Solaris as a platform for Lotus Notes and at home in his spare time. He currently lives in San Francisco and can be reached at dwarf333@hotmail.com.

Paul A. Watters is the project manager at Neuroflex, where he's responsible for developing natural language database systems in Java on the Solaris platform. 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 journals@mail.training.com.

Editorial

Editor	Garrett Suhm
Assistant Editor	Jill Suhm
Managing Editor	Joe Froehlich
Copy Editors	Rachel Krayer
	Christy Flanders
Contributing Editors	Clayton E. Crooks II
	Edgar Danielyan
	Alan Orndorff
	Paul A. Watters
Print Designer	Melissa Ribaudo

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 inside_solaris@zdjournals.com.

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

ZD Journals

General Manager	Kelly Baptiste
Manager of Customer Relations	Heather Loaicono
Manager of Operations	Cristal Haygood
Manager of Print Design	Charles V. Buechel
Manager of Product Marketing	Mike Mayfield
Product Marketing Manager	Brian Cardona

Postmaster

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

Postmaster: Send address changes to

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

Price

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

Our Canadian GST# is: R140496720. CPM# is: 1446703. QST# is: 1018491237.

Back Issues

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

Are you moving?

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

Coming up

USPS ARMIN PSI 881 APPROVED POLY

- Network Security
- Using MD5
- Free Enterprise Javabean Servers

PERIODICALS MAIL

2096

**************3-DIGIT 480
C: 7661905 00002096 04/00
28
66
[redacted]

Please include account number from label with any correspondence.

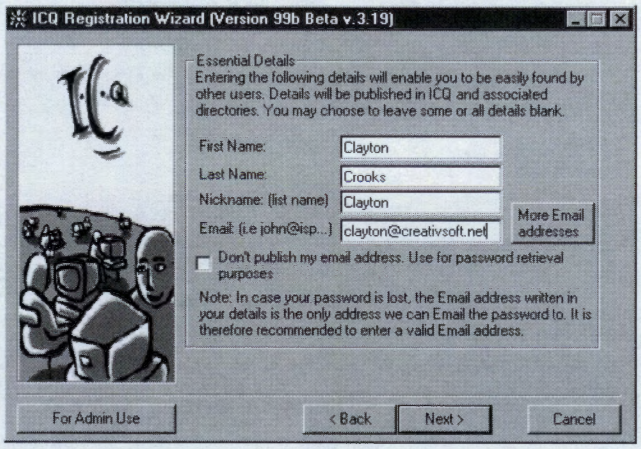
troubles. You should not install the software at this time, as there are a few items that need to be addressed first.

The second step is to download and install the Solaris JDK (Java Developer's Kit) if your system doesn't already have a compliant version. Version 1.1.3 or greater is required, although we would recommend downloading and installing the most recent version (1.2.1_03 at the time of writing). You must perform this step because there isn't a JRE (Java Runtime Environment) included with the downloaded software. After downloading and installing the JDK, which you can obtain from **www.sun.com**, you can now begin the ICQ installation.

Installing ICQ

The installation is fairly simple and should take only a few minutes. You can complete it by following the associated prompts. After the installation is complete, the next step is to register with ICQ. Before registering, you must first connect to the Internet. Registration is quick and simple. An onscreen registration form shown in **Figure B**, pops up, when you launch the program for the first time.

After filling in the appropriate personal information, you'll receive an ICQ #. This number (along with a secret password) identifies you as a registered ICQ user. You can publish your ICQ number on your Web site or business card so that other ICQ users can easily identify you.



Essential Details
Entering the following details will enable you to be easily found by other users. Details will be published in ICQ and associated directories. You may choose to leave some or all details blank.

First Name: Clayton
Last Name: Crooks
Nickname: (list name) Clayton
Email: (i.e john@isp...) clayton@creativsoft.net

Don't publish my email address. Use for password retrieval purposes

Note: In case your password is lost, the Email address written in your details is the only address we can Email the password to. It is therefore recommended to enter a valid Email address.

Figure B: *You register with ICQ with this dialog box.*

Using ICQ Java

After the initial registration, using the basic features of the software is quite simple. Launching ICQ Java and connecting to the Internet will automatically open the contact list of the last ICQ user to have used the computer. If you didn't select the Save Password check box during the registration process, you'll be prompted by a password entry/verification window. Type in your password to log on. If you did check the Save Password check box upon registration, the last ICQ window used will automatically appear.

You're now ready to use all of the features of the ICQ Java program. Many tutorials exist on the use of the software. You should also read the help and tutorial files that you downloaded with the ICQ Java package. The ICQ home page, **www.icq.com**, has excellent message boards for posting questions regarding the installation or use of the software.

ZD Journals Contacting Customer Relations

If you have questions or concerns about your subscription, you can contact our Customer Relations Department by sending email to **journals@mail.training.com**. You can also contact us by phone at (800) 223-8720. Be sure to include or have on hand your customer number when you contact us. Doing so will help us to quickly and easily assist you.