

# Inside Solaris™

Tips & Techniques for users of Sun Solaris

## Installing and configuring Webglimpse

by Alvin J. Alexander

If you're a system administrator or Webmaster for one of the Fortune 1000 companies in the world, the cost of a powerful search engine tool for your Internet/intranet Web sites may not be a major concern. But if you're a bit smaller, or you're concerned about paying a vendor based on the number of documents you index, you might want to read on.

Recently, I ran into a search engine tool for Web sites at the University of Arizona that caught my attention. The search engine they created is called Webglimpse, and I've been impressed by what I've seen. Webglimpse is used by Web sites around the world for their search needs. This includes organizations like the American Cancer Society ([www.cancer.org](http://www.cancer.org)) shown in Figure A.

### What's Webglimpse?

Webglimpse is a search engine for Internet, intranet, and extranet Web sites. It offers some of the same search engine features you'll find at Internet sites like AltaVista and Excite—at a fraction of the cost of these well-known commercial programs.

Webglimpse is (primarily) a Perl/CGI program that accepts input from HTML forms. It interacts with Glimpse indexes to locate the documents you

might be interested in. *Glimpse* is a command-line tool for Solaris, also provided by the University of Arizona. Glimpse lets you perform lightning-fast searches from the Solaris command line, and Webglimpse extends that capability to the Web.

To Web site visitors, your Web pages can contain search forms that look similar to AltaVista or Excite forms. Just design your HTML pages as you normally would, and then add a variation of the Webglimpse search form wherever you want it. If you're interested in creating a search engine that spans multiple Web sites, Webglimpse also offers the ability to traverse remote Web sites.



### In this issue:

1 Installing and configuring Webglimpse

7 Lexical and text analysis

8 Tool of the Month: IsIK

9 Sun/UNIX editors

10 SPARC OpenBoot and the Forth language

12 Sun-provided tool to test Year 2000-compliance

14 Quick Tip: Configuring network printers in Solaris 2.6

16 Ping the Solaris Dude:

- What version of BIND am I running?
- What's a segmentation violation?

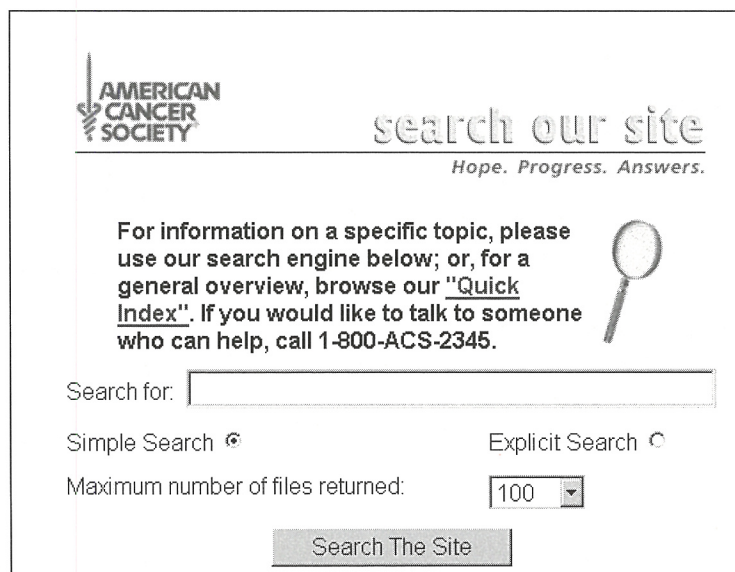


Figure A: Here's Webglimpse in action at the American Cancer Society.



## Features

In addition to its low price, I find several things about Webglimpse attractive. Its special features include the following:

- Webglimpse can account for misspellings (up to two characters) in your searches.
- When Webglimpse finds documents that match your search, it can provide hyperlinks directly to the lines of returned documents that match your search pattern.
- As an administrator, you can specify the names of documents to exclude from searches (for example, documents like \*.old or \*.bak).
- Because you have the source code for the CGI program, you can modify the format of your output pages.
- You can specify the maximum number of items returned.
- You can quickly add search boxes to all of the Web pages in a directory tree.
- You can specify the maximum number of characters printed per match.
- It will return only recent files.
- It's configurable for multiple domains on one server.

Webglimpse also offers several traditional search engine features:

- Traditional keyword searching
- Partial word matches
- Case-sensitive searching
- The use of Boolean queries (AND, OR, NOT word combinations)

## Limitations

While Webglimpse does many things very well, it doesn't provide a *relevancy factor*, or *weighting*, of the pages it returns. Most search engines, like Infoseek, return from a search and rank documents from 100 percent relevant to 1 percent, and return the documents in that order. (Of course, how they determine what's relevant is another story.) Webglimpse doesn't

offer that capability. From what I've seen, this appears to be its most severe limitation.

Having said that, this limitation is partially overcome by the way Webglimpse prints the lines from the documents that match your search criteria. While traditional search engines print a paragraph from a document's "description" META tag, Webglimpse prints the exact lines that are matched by your search criteria.

With this feature, you can see the matching lines in the document, and determine the relevancy of the match yourself. It just depends on whether you prefer to see weighting factors or the actual document text.

## Requirements

Webglimpse has a few requirements that you should be aware of before proceeding with an installation. First, Webglimpse is primarily a Perl program, so you'll need to have Perl installed, specifically Perl 5.0 or newer.

Second, Webglimpse requires that Glimpse be installed first. Specifically, Webglimpse 1.6 requires Glimpse v4.1 or newer to be installed first.

Third, you'll need a C compiler—but only during the installation. When you install Webglimpse, it compiles a couple of small programs based on the environment settings you provide. The standard C compiler or GNU compiler should work just fine.

## Downloading and installing

To use Webglimpse, you'll need to download it, install it, and then configure your first archive. Webglimpse can be downloaded from these URLs:

- <http://tucson.com/webglimpse>
- <http://donkey.cs.arizona.edu/webglimpse/>

The [tucson.com](http://tucson.com) URL is now the preferred URL, because the Internet WorkShop group provides support for the product. Just look for the Webglimpse download section, where you can download the tarred and gzipped file.

After I downloaded the software, the installation was relatively easy, but there were two problems worth mentioning. First, make sure you meet the mentioned requirements. These requirements aren't explicitly stated in the installation documentation.

Second, the `Makefile` script for Solaris assumes that you'll be using GNU's `gcc` compiler on a Solaris platform. If you're using Sun's `cc` compiler, you'll want to edit the `Makefile` before running `wginstall`, or else the install program will bomb and you'll need to start over.

Everything else seemed to follow the installation instructions closely. When you're finished, Webglimpse will be installed in a directory such as `/usr/local/webglimpse`, and a few utility programs will be installed in `/usr/local/bin`.

In summary, the installation procedure could be a bit smoother, but it's already much better than the earlier procedure for Webglimpse 1.5. For me, the installation was the hardest part about using Webglimpse.

## Creating a local archive

Once you have Webglimpse installed, you're ready to create your first archive. You can create this archive when you're prompted at the end of the installation procedure, or wait until a later time. For the purposes of this article, I suggest that you create your first archive in just a few moments.

Any time you want to create a Webglimpse archive, you'll run a Perl program named `confarc` (short for configure archive), which is located in the Webglimpse directory `/usr/local/webglimpse`, by default. The `confarc` program prompts you with a series of questions, and creates an archive based on your replies. It also generates a sample HTML search page that you can use to test your archive, and include in your own HTML documents.

For the remainder of the discussion, I'll need to refer to a sample directory and a sample URL on a Web server. The sample directory I'll refer to is the Apache Web server's HTML user's manual directory:

```
/usr/local/lib/apache/htdocs/manual
```

I'll use these manuals as an example of a group of HTML documents that you might want to index for later searching. On my intranet, the URL for the manual directory is:

```
http://intranet.missiondata.com/manual/
```

With this as a sample directory, we're ready to proceed. Move to this directory, and run the `confarc` program:

```
cd /usr/local/lib/apache/htdocs/manual
/usr/local/webglimpse/confarc
```

The `confarc` command prompts you with several questions you need to be prepared to answer. In this first example, we'll just create an index of local files, so we'll only need to answer these four questions:

1. In what directory do you want to store the Webglimpse-generated index files?
2. What title do you want to put on the archive?
3. Do you want to index by (D) directory or (T) traverse URLs?
4. What's the full path of the directory to be indexed? (Must be accessible from the Web.)

For the purposes of my intranet server and the manual directory, the answers for these four questions are:

1. `/usr/local/lib/apache/htdocs/manual`
2. Index of the Apache manuals
3. `d`
4. `/usr/local/lib/apache/htdocs/manual`

As fair warning, be careful *not* to select `t` in step three. If you enter `t`, things start to get very interesting, because this gives you the power to index remote and Web servers, which we don't want quite yet.

Once you've answered these questions, `confarc` builds the index files you need. In fact, it creates a large number of files, with most files matching `.wg*`, `.g*`, and `wg*` metacharacter patterns. The process also creates another file named `archive.cfg`, and a directory named `.remote` (for indexing remote sites). Out of all these configuration files, **Table A**, on page 4, shows a list of the most important files generated by `confarc`, with brief descriptions of each file.

The `confarc` procedure also generates two sample HTML search forms you can use, named `wgindex.html`, and `wgall.html`. These files will also be located in the manual directory (or whatever directory you chose to index).

The first HTML page, `wgindex.html`, includes the standard Webglimpse search form. It lets you specify:

- The text string(s) to search for
- Whether the search should be case-sensitive



- Whether you want to match partial words
- Whether you want to jump right to the lines in the files matching your search

The second page, `wgall.html`, lets you add a little more precision to the search, because it lets you specify the directory tree you want to search via a dropdown scrolling list. This can be very good for power users who know

**Table A:** The most important files generated by the `confarc` program

File	Description
<code>archive.cfg</code>	Defines the archive configuration parameters for Webglimpse. Built from the questions asked by <code>confarc</code> .
<code>.wgfilter-index</code>	Defines the files that you do and don't want to index.
<code>.wg_err</code>	A list of error messages.
<code>.wg_log</code>	A list of all collected files and URLs, including files indexed and excluded.
<code>Wgreindex</code>	An auto-generated command file that can be used to build the archive and index. You can put this command in your cron file to rebuild the index periodically.
<code>.wgsiteconf</code>	Configuration information about your Web site.

what's in each directory tree, but might not be good for people that don't know where the desirable documents may be located.

## Testing Webglimpse

Once you've generated the index files and search forms with `confarc`, all you need to do to test the installation is point your Web browser to the proper URL. In the case of my server, the proper URL is:

`http://intranet.missiondata.com/manual/wgindex.html`

When you point your browser at this URL, you'll see a form similar to the one shown in **Figure B**. To test the installation, just type in the name of a word that should be in the archive and click the Submit button. In the case of the Apache online manuals, I searched for strings like `srm.conf` and `DocumentRoot`. (If you're interested, I also recommend selecting the Jump To Line option.)

If everything works okay, your search results will be displayed in the next page. You'll see results similar to those shown in **Figure C**.

As you can see from **Figure C**, turning on the Jump To Line feature of Webglimpse makes the search even more powerful. If you see a line you're interested in, you can just click that line. Webglimpse takes you directly to the line inside the document you select.

## Power searching

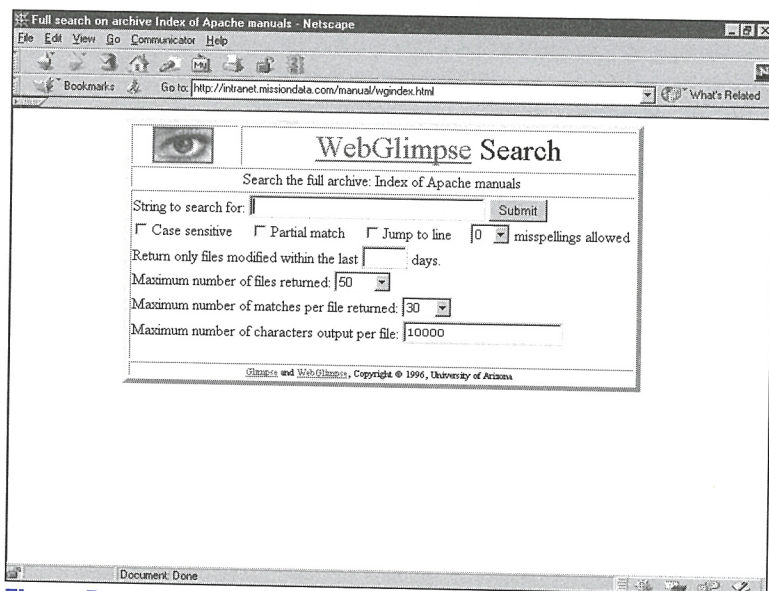
The options provided on the auto-generated HTML forms (case-sensitive, partial match, etc.) are very powerful, and for the most part, self-explanatory. If you don't know what they mean at first, you'll learn very quickly with a little experimentation.

Some of the other search features that Webglimpse offers aren't immediately obvious. With Webglimpse, I've found that you can combine words in your searches using the Glimpse command-line syntax.

For instance, if you want to perform searches on a combination of words, using a Boolean AND operation, just connect your search keywords with a semi-colon (;).

As an example, if you want to search for documents containing the strings `srm.conf` and `DocumentRoot`, just enter this syntax in the search box:

```
srm.conf;DocumentRoot
```



**Figure B:** The file `wgindex.html` contains the standard Webglimpse search form.



**Note:** Be careful not to include spaces in the search pattern that you don't really want. Spaces will affect your search results.

If, instead, you want to search for documents containing the strings `srm.conf` OR `DocumentRoot`, just use the comma (,) instead to separate the words:

```
srm.conf,DocumentRoot
```

I really like these features, and I'd recommend including references to them on your search forms. I suppose it's probably more standard to support keywords like AND and OR in search engine tools. If you're a sharp Perl programmer and you're interested in supporting these words instead, it won't take much to code this change, if you're interested. I suspect the comma and semi-colon characters are left the way they are to give the end user more precision in their searches.

## Running a periodic index with crontab

If you have an active Internet or intranet Web site, you'll probably want to run the index program, `wgreindex`, fairly often, so new files will be added to the index automatically. You can easily set this up by adding a line to your crontab file for each index you want to regenerate.

For my purposes, a line similar to this works just fine:

```
55 11 * * * /usr/local/lib/apache/  
→htdocs/manual/wgreindex -q > /dev/  
→null 2>&1
```

If you prefer to retain the `wgreindex` output, just redirect the output streams to your desired output files.

## Indexing remote Web sites

If you're interested in creating a multi-site search engine, one of the great things about Webglimpse is the ability to index documents contained on other Web servers. This capability can quickly turn your Web server into a small version of AltaVista or Excite. (From a technical perspective, I don't refer to Yahoo!. Technically, most of what Yahoo! offers is a directory, which is different from a document-based search engine.)

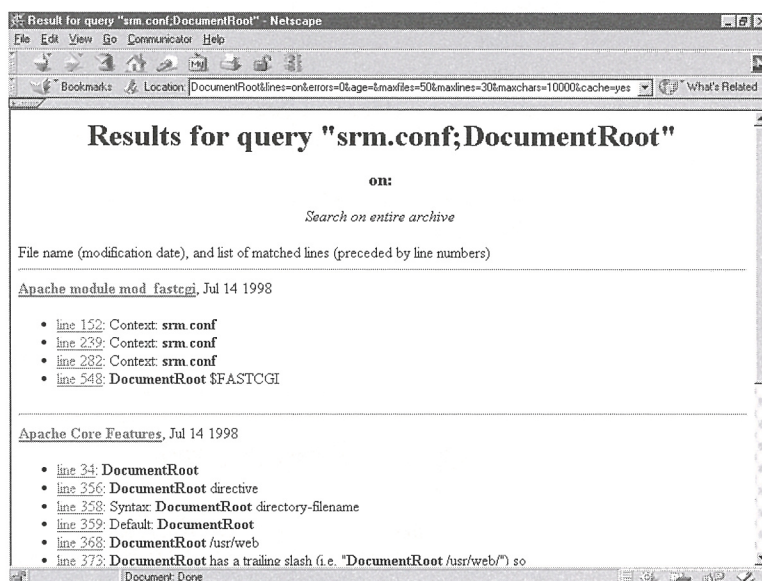
Before attempting to index any remote sites with Webglimpse, I suggest that you first print the support document titled "Configuring an archive." This document contains a description of each question you'll be prompted for, and it will be very helpful.

Also, you may need to create a document on your local Web server that contains hyperlinks to the remote sites you want to index. For instance, let's assume that you want to index two remote sites named `coyote.acme.com` and `bugs.bunny.com`. In this case, you'll need to create a document on your server with entries similar to these:

```
<a href="http://coyote.acme.com">coyote</a>  
<a href="http://bugs.bunny.com">bugs</a>
```

If you don't already have a document with remote links like this, you should create it, and save it with a suitable name—perhaps something like `remote-sites.html`. For lack of a better term, you're going to *feed* this document to `confarc` in just a few moments. As a final note, be sure to save this document in a location where it can be accessed via a valid URL. I saved this document in a directory named `/usr/local/lib/apache/htdocs/remote`.

With this feeder document in place, you're now ready to begin the process of indexing these remote Web sites. If you're ready, move to the directory containing your feeder document, and start the `confarc` program just as you did before:



**Figure C:** The search results page demonstrates the Jump To Line feature that Webglimpse provides.



```
cd /usr/local/lib/apache/htdocs/remote/  
usr/local/webglimpse/confarc
```

Answer `confarc`'s prompts as usual, until it asks, "Do you allow traversal of remote pages?" Previously you answered *n* at this point, but now you want to answer *y*.

With this reply, `confarc` begins prompting you with a series of seven additional questions. This is where it helps to have the printed document, "Configuring an archive," in hand.

I'll skip the discussion of the first six prompts, because their meaning is fairly well-defined in the "Configuring an archive" document.

The seventh prompt looks like this:

Now you'll need to enter the URL(s) of the file(s) you'd like to traverse:

This is where you should supply the URL to your `remote-sites.html` feeder document. For my setup, the proper URL is:

```
http://intranet.missiondata.com/remote/  
remote-sites.html
```

At this point, Webglimpse prompts you for the next URL. Here you can keep entering URLs until you're finished, or hit [Enter] when you have no more URLs to provide. If the `remote-sites.html` document is the only document you want to refer to, just hit [Enter].

Now, `confarc` will begin its indexing process. Depending on the URLs you provided, the options you selected, and the speed of your network, this can be fairly quick, or quite lengthy.

When `confarc` finishes, you can test the archive just as you did before, by pointing your browser to the proper URL on your server:

```
http://intranet.missiondata.com/remote/  
wgindex.html
```

Then, enter a few keywords to search for. If everything succeeded, you should be able to find keywords in remote documents just as easily as you find them in your local documents.

## Cost issues and other free (or nearly free) search engines


The price of Webglimpse currently varies from free to the high price of \$2,000, based on your use. For government and educational use, it's free; for small businesses, it's available for as little as \$200. At the high end, the largest businesses will be charged \$2,000, much less than the cost of well-known search engine products. I'd suggest looking through the Webglimpse URLs provided earlier for pricing for your intended environment.

If you're on a tight budget but interested in incorporating a search engine into your Web sites, and Webglimpse doesn't meet your needs, here are two other search tools that are free or nearly free:

- Webinator, from Thunderstone: [www.thunderstone.com/webinator](http://www.thunderstone.com/webinator) (free for small sites)
- ht://Dig, located at [www.htdig.org](http://www.htdig.org)

I haven't personally used these products, but noticed their pricing strategies while researching this article.

## Conclusion

With Webglimpse, you can add a fast, low-cost, document-searching engine to your Internet or intranet Web servers. Webglimpse can be used to serve documents from your local Web site, as well as remote servers, so you can quickly start your own search engine site and compete with AltaVista, Excite, and others. 

## Questions? Comments?

Is your Solaris box singing the blues? Poor performance got you down? Ping the Solaris Dude today! Send your questions to [robt@cymru.com](mailto:robt@cymru.com). See you next month and keep those questions coming!



# Lexical and text analysis

by Paul A. Watters

Every day we seem to find situations that require manipulating text. This may be as simple as extracting email addresses from messages to massaging data that needs to be fed to a supercomputer for analysis. In this article, we'll look at utilities to make text processing on Solaris easier.

The processing of text and lexical information was one of the earliest non-numerical uses for digital computers, and has continued to be one of the most popular. But text analysis goes beyond simple word processing, especially with the preparation of text for some kind of data analysis (that is, database entries or statistical observations). Fortunately, Solaris is well equipped to assist many text processing applications. Rather than re-inventing the wheel for text processing, an optimal approach is to combine custom tools with existing utilities on the command line to perform complex lexical tasks.

## Text utilities

We'll begin by looking at some of the basic text processing utilities available under Solaris. We're all familiar with basic commands like `cat`, which sends the contents of a file to standard output, or `more`, which presents the display of standard output in readable segments. These commands can be combined on the command line with a pipe (`|`) to increase their functionality:

```
cat filename | more
```

The same results could be achieved by using the redirection operator (`<`) to retrieve the contents of the file, instead of using `cat`:

```
more < filename
```

However, if `filename` was a list of simple database entries from a pet shop, which either contained the field `bird` or `animal`, a more complex query could be formed to retrieve all the records for `animal` page by page, by incorporating the `grep` command:

```
cat filename | grep "animal" | more
```

If we only wanted to sample the first few lines of `filename` (which might be the case if a non-exhaustive search is being performed), we could use the `head` utility:

```
head filename | grep "animal"
```

Alternatively, if the database was sorted by `animal`, and we were interested in finding the number of `maltese terriers`, we could use the `uniq` utility (with the `-c` flag). This will print the first occurrence of a unique line of text in a file with respect to adjacent lines, and count the number of times it occurs:

```
uniq -c filename
```

If we simply want to count the number of lines and the total number of characters, we can use the `wc` utility:

```
wc filename
```

If the database fields were unsorted, there's a very useful utility called `sort` that can sort in dictionary order, or can convert lower case to upper case and sort, and even has a built-in mechanism to sort by three-letter date codes.

## Simple tools

We examine a simple application that combines the text processing utilities outlined above with several small utilities written in C. The task is to process a text document and create a numerical representation of the data that's suitable for statistical analysis. This involves creating an index of unique terms in the document, which are then substituted for each word in the original document to form a set of numerical vectors that can be read into a statistical package. This situation arises quite often in information retrieval applications, where a database of unique definitions and terms is created from a set of documents. The first custom program, `listwords`, reads in a document (`input.txt`) and creates a text file (`output.txt`) containing all the words in the document

```
listwords input.txt output.txt 5000
```



Next, we sort the `output.txt` file and generate a list of unique entries:

```
sort -u output.txt > unique.txt
```

At this point, non-words could be excluded by using the `spell` utility, if we were only interested in processing words found in the dictionary. We then verify the number of unique entries in the list by using the `wc` utility:

```
wc unique.txt
```

The result (in this case, 500) is then used as a parameter for the second custom program, `createindex`, which simply appends an index to each entry in the file:

```
createindex unique.txt index.txt 500
```

The unique numerical types defined by the index are then inserted into a corresponding position, with an output file generated by the third custom program:

```
tokenise index.txt input.txt output.txt
```

Although separate programs or functions could have been written as replacements for `wc` and `sort`, the great advantage of using separate command-line utilities is that a new program doesn't have to be written to perform slightly different text processing tasks. In addition, new utilities can be written to extend the functionality of those included in the Solaris distribution. The flexibility of these utilities can also be combined with C-shell commands, such as `foreach`, to process many files at one time. This is a particularly important feature that limits graphical-user interface text processing systems.


## Complex tools

Sorting and indexing are just the beginning of processing text with Solaris. There are more complex tools for generating lexical analyzers and parsers (for example, `lex` and `yacc`). `Lex` is best used for simple pattern matching tasks, and can also be used to assign unique types to words, as we've demonstrated, using several different utilities. While `Lex` is more compact than using several utilities to perform the same task, it has the drawback of a difficult syntax consisting of definitions, rules, and routines, which is less intuitive than dividing a complex task between dedicated utilities. `Yacc` also has similar problems, but nicely handles the processing of complex lists by using recursion. `Awk` is yet another useful pattern-scanning utility, for which associations can be made between a regular expression (that is, using Boolean operators) and some action (that is, splitting and truncating text).

## Conclusion

Solaris has many tools available for text processing, from simple command-line utilities that can sort and process text information from databases, to sophisticated lexical analysis programs. A suite of tools can be constructed to handle many different scenarios without having to rewrite a new program for each task. `Lex` and `Yacc` aren't for the faint-hearted, but worth learning if routines are required for resolving ambiguities using precedence rules.

## Further reading

As usual, the O'Reilly series of books has a useful manual for `Lex` and `Yacc` by John Levine. The custom utilities described above can be obtained via email from the author. 

## Tool of the Month: `lslk`

The tool of the month is `lslk`, or list-locks. This tool, created by Vic Abell, is a great way to list all the system locks. Ever wonder what has a file locked up and just won't let go? With `lslk`, you'll know!

`lslk` displays the inode and filesystem of the locked files. With the wealth of options to `lslk`, you can avoid kernel deadlocks and customize the search parameters. `lslk` also

tracks NFS-based locks. One downside: The VxFS file system isn't supported.

You can obtain `lslk` at the following URL  
<ftp://vic.cc.purdue.edu/pub/tools/unix/lslk/>

I don't warrant this tool or its use. However, should you encounter any problems porting or using the tool, don't hesitate to drop me a line.



# Sun/UNIX editors

by Werner Klauser

There are several different editors available on the UNIX systems. Some of these editors are designed to run under X-Windows, while others will run on any terminal. This article explains how to open each editor, make a few changes, and then exit the editor. You can consult the editor's man pages for additional information. All of these packages are full screen editors. Each can be invoked by typing

```
$ edit myFile.txt
```

where *edit* is the editor that you want to use and *myFile.txt* is the file that you want to edit. When you are using X-Windows, you can select many of these editors from the Editors | Word-processing submenu that pops up when you right-click.

## Character-based editors

While it seems that the world has gone completely graphical, there still is a place for character-based editors. Following are some of the more popular editors available for Solaris.

### vi

The vi editor is the basic all-purpose text editor. There are two modes in vi: one for entering or changing the file and the other for command execution. You begin the session in Command mode. To switch to Editing mode, you can type *i* to insert before the current mark, *a* to append after the current mark, and *x* to delete a character. If you precede *x* with a number, then that number of characters is deleted, starting with the current mark. To switch back to Command mode, press [Esc]. In Command mode, use the arrow keys to move around until you find something that needs to be changed or added; then type *i*, *a*, or *x* to make changes. You can also save your file while in Command mode with *:w*. If you want to quit, you can use ZZ to save the current file and quit or *:q!* to quit without saving.

### emacs

The emacs editor is much more powerful than vi. When running under X-Windows, you can use the mouse to place the insertion point any-

where in the window; otherwise, just use the arrow keys. Along the bottom of the emacs window is a status bar, which provides you with some useful feedback. Many emacs commands require two key strokes. If you accidentally begin a command, you can press [Ctrl]G to cancel it.

There's a helpful tutorial that you can access by pressing [Ctrl]HT. You can save your work by pressing [Ctrl]X [Ctrl]S. To quit emacs, press [Ctrl]X [Ctrl]CL.

If you haven't saved changes, then you'll be prompted to do so. You can also cut and paste portions of text easily with emacs. To do this, you first set a mark at one end of the text that you want to cut by pressing [Ctrl]2. Then you move to the other end of the text you're cutting and press [Ctrl]W. To paste the block of text that you've just cut, press [Ctrl]Y. Every time you press [Ctrl]Y, a copy of the text that you cut is inserted.

### pico

The pico editor, like emacs, can be used from either a terminal or X-Windows. To make changes to a file, use the arrow keys to move around. Once you've gotten to the place where you want to insert something, start typing. You can delete text with the [Backspace] or [Delete] keys. The different commands that pico supports are displayed along the bottom of the window, each preceded with a caret (^). The caret indicates that you should hold down the [Ctrl] key while pressing the command key. For instance, the exit command is ^X—this is equivalent to [Ctrl]X.

If you're looking for a simple but powerful editor to learn, then pico is probably the one for you. It has a helpful interface and can be used from both X-Windows and a regular terminal.

### editor

Like pico, editor has a good interface and works well from a terminal. You can use the arrow keys to move the insertion point and the [Backspace] or [Delete] keys to delete text. The supported commands are listed in the top half of the window and have a syntax similar to pico's. There are some two-keystroke commands. For instance, to move to the top of a



file, the command is `^KU`, which is equivalent to `[Ctrl]KU`. The important commands are all displayed, so you shouldn't have any trouble learning this package.

## X-Windows editors

Following are some of the more popular X-Windows editors available for Solaris.

### textedit

The `textedit` editor is an X-Windows-only editor. Like the other X-Windows editors, `textedit` has a helpful graphical user interface. Once you've started it, you can make changes to your file. `textedit` has mouse support so you can use the mouse to place your cursor and scroll through your document. To exit `textedit`, right-click on the window's title bar. This will bring up a menu for that window. Move the mouse down to the Destroy Window option to quit. Be careful not to select Exit Window Manager from the menu, as this would log you off of the machine, in addition to quitting `textedit`.


### aXe

The `aXe` editor is also an X-Windows editor. Although the other packages support similar features, the graphical user interface makes `aXe`

easy to learn. After starting it, you should see two windows—one for your document and one with `aXe` written in the middle of it. Your document window contains the text of your current document, while the `aXe` window provides easy access to the Online Help feature. Simply click on the Help button to get more help.

Like the other packages, you can use the arrow keys to move around. `aXe` also supports searching and cutting and pasting. To cut a block of text, select the text with your mouse (the selected text will be highlighted). Now, from the Delete menu, select Cut. This stores the selected text in a buffer. To paste the text that you just cut, select Insert | Paste. You can save your changes and/or quit using the different options in the Quit menu.

### xedit

The `xedit` editor is another X-Windows editor. It isn't as powerful as `aXe`, but the interface is easier to learn because of its simplicity. You can cut and copy as you would in the UNIX shell using the mouse buttons. For instance, select the text that you want to copy, and move to the place where you want to insert the text. Press the middle mouse button to insert the text. There are buttons across the top of the window for saving changes, loading files, and quitting. 

# SPARC OpenBoot and the Forth language

by Boris Loza

From time to time as a UNIX system administrator, I've had to work in the Solaris OpenBoot environment. It's useful for booting the operating system (`boot -r`, `boot cdrom -s`, etc.), modifying system start-up configuration parameters (`input-device`, `output-device`, `setenv`, etc.), troubleshooting (`probe- scsi-all`, `show-devs`, etc.), or running diagnostics (`test net`, `test /memory`, etc.). But sometimes, it isn't enough to use predefined commands and utilities. For this purpose, OpenBoot provides a very powerful environment based on the ANS Forth programming language.

## Some Forth history

The name Forth was intended to suggest software for the fourth (next) generation computers, which Charles Moor (the programmer who invented it) saw as being characterized by distributed, small computers. The operating system he used at the time restricted filenames to five characters, so U was discarded. The first Forth interpreter was written in 1968. For the next five years, Forth was implemented on various CPUs and became widely known because of its high performance and economy of memory.

In 1988 Sun Microsystems invented Open Firmware technology—hardware-independent boot code, firmware, and device drivers. Open Firmware, then called OpenBoot, allows one version of the Boot ROM to run on any configuration of hardware and software. Such technology uses Forth as the official language.

## Why Forth for all this?

Forth is a stack-based, extensible language without type-checking. It uses “reverse Polish” (postfix) arithmetic notation, familiar to users of HP calculators. To add two numbers in Forth, you’d type `2 3 +` instead of `2 + 3`. If you’re using Forth, you don’t need to recompile your program to add new functionality. You can define a new command and it will instantly be available for you to use. Because of this, the Forth compiler is simpler, smaller, and faster than other compilers. So the interactive Forth system, including an editor, assembler, and even multitasking support, can easily be put in an 8K EPROM!

## The Forth language

The fundamental program unit in Forth is the *word*—a named data item, subroutine, or operator. Actually, OpenBoot commands such as `boot`, `printenv`, and `probe-scsi-all` are Forth-defined words. Programming in Forth consists of defining new words in terms of an existing one.

You can start programming in Forth at the OpenBoot ok prompt (ok is the usual prompt in Forth):

```
ok : average ( a b - avg ) + 2/ . ;
ok 10 20 average
ok 15
ok : .ASCII ( end start -- ,
    dump characters )
    do
        cr i . i emit \ Print ASCII
        characters
    loop ;
ok 70 65 .ASCII
ABCDEF
ok
```

OpenBoot 3.x contains about 2,450 Forth words. All words belong to the dictionary that also contains vocabularies (consisting of related words and variables).

The new commands created above would be lost after rebooting a machine. OpenBoot

provides a way to prevent this by saving into NVRAM using `nvedit`:

```
ok nvedit
0: : hello ( -- ) cr
1: ." Welcome to OpenBoot!" cr ;
2: ^C
ok nvstore
ok setenv use-nvramrc? true
ok reset-all
--
ok hello
Welcome to OpenBoot!
ok
```

By creating customized scripts, you can modify the OpenBoot start-up sequence. Unfortunately, you can’t use the following commands here: `boot`, `go`, `nvedit`, `password`, `reset-all`, and `setenv security-mode`. OpenBoot provides various facilities for debugging Forth programs and loading and executing programs written in Forth from Ethernet, a hard disk, or a floppy device.

One of the Forth utilities that we’d like to mention here is the built-in Forth language decompiler—`see`. It can be used to re-create the source code for any previously defined Forth word. For instance:

```
ok see scan-subtree
: scan-subtree
['] scan-subtree guarded-execute drop
;
ok see probe-scsi-all
(ffd60c5c) ['] (ffd88b08) scan-subtree
;
```

The preceding listing shows that `scan-subtree` is composed only of Forth source words that were compiled as external or as headers with `fcode-debug?` set to true. `probe-scsi-all` is a different word. It also contains words that were compiled as headerless and are, consequently, displayed as hex addresses surrounded by parentheses. For more information on how to use Forth development tools, consult the OpenBoot reference manual.

## Forth and shell scripting

On the Internet, you can find various shareware ANS Forth compilers for a number of Operating Systems. One of the most interesting is pForth (a portable ANS style Forth). After compiling and linking it on `/usr/local/bin/forth`, you can run standalone scripts like




## Listing A: A simple Forth script that can run on Solaris

```
#!/usr/local/bin/forth
\ ////////////////////////////////////////////////////////////////////
\ Enter a number:
\ 1. Item number one
\ 2. Item number two
\ 3. Item number three
\
\ 0. Exit
\ ////////////////////////////////////////////////////////////////////
: get-menu-item ( -- n ) \ Query user for a menu option.
begin
  ." Enter a number: " cr
  ." 1. Item number one" cr
  ." 2. Item number two" cr
  ." 3. Item number three" cr cr
  ." 0. Exit" cr
  key dup emit cr ascii 0 - dup 0 3 between not
while
  drop
  ." Not a valid menu number! Try again." cr cr
repeat
case ( case-value -- )
  1 of ." This is number one"      endof
  2 of ." This is number two"      endof
  3 of ." This is number three"    endof
  0 of ." Exit!"      bye          endof
endcase ;

get-menu-item
```

the one shown in **Listing A**. To run this program, just type the name of the file.

get-menu-item

In our opinion, Forth cannot replace Perl and UNIX shell programming facilities for System Administration needs. These languages are specially designed for parsing strings and I/O manipulation. But you can practice with Forth scripting in order to be more comfortable with creating power tools in the OpenBoot environment. 

## References

1. *Scientific FORTH: a modern language for scientific computing*, Julian V. Noble, Mechum Banks Publishing, 1992, 300 pages, ISBN: 0-9632775-0-2, disk included. (This book contains many serious examples of Forth programming style and useful programs.)
2. *Forth: The New Model—A Programmer's Handbook*, Jack Woehr, M & T Publishing, 1992, 315 pages, ISBN: 0-13-036328-6, DOS disk included. (Describes features of ANS Forth and how to use it to write Forth programs. Currently the only book about ANS Forth.)
3. Forth Interest Group Home Page: [www.forth.org/](http://www.forth.org/)
4. Forth on the Web: <http://pisa.rockefeller.edu:8080/FORTH/>

# Sun-provided tool to test Year 2000-compliance

by Werner Klauser

The millennium is fast approaching and there's a growing urgency to test your system for Year 2000-compliance. You want to know if there's any source code or tool that Sun can provide to help you do the testing of all the software, including third party software, on all the Solaris boxes in your organization.

## Download it

You can download an ABI (Application Binary Interface) tool, called y2000, free of cost from Sun's Web site at


[www.sun.com/y2000/abi-download.html](http://www.sun.com/y2000/abi-download.html)

Once you install the tool on your machine, you can test any binary executable for Year

2000-compliance. The tool tests any binary application and all the dynamic libraries it depends on for dates and reports if it passes the 2000 bug. Since it's an ABI tool, it can be used on any application, including third-party software that runs on Solaris. Be aware that to be absolutely sure of Year 2000-compliance, it's necessary to do a final human testing of the source code. This tool is definitely an important starting point. As an example, using

the y2000 tool to test the system's /bin/ls program results in the output in [Listing A](#).

## Summary

From the tool's output, it's quickly noticed that the y2000 tool searches for system library functions that could have potential Year 2000-compliance problems. You then must carefully ask yourself whether this potential problem is an actual problem. 

### Listing A: Example output from y2000 tool

---

This report singles out your application binaries that have dependencies on time related functions.

Improper use of these functions may cause a Year 2000 transition problem, but the dependency alone does not signal a Year 2000 error: manual source code checking and/or transition simulations are the next required steps.

For additional information on developing Year 2000 safe applications, please see <http://www.sun.com/y2000/devguide.html>

Function	Dependency	Report
-----	-----	-----

These time related functions are checked for:

asctime	asctime	asctime_r	ctime	ctime
ctime_r	difftime	getdate	gettimeofday	gmtime
gmtime_r	localtime	localtime_r	mktime	settimeofday
strptime	strptime	time	tzset	tzsetwall
utime				

The following applications have a dependency on one or more of the above functions, your next step should be to examine them more closely for Year 2000 transition problems:

```
/bin/ls:  
  ctime time
```

1 of your applications out of a total 1 have a dependency on one or more of the time related functions listed above.

The complete list of applications examined was:

```
/bin/ls
```

Note that shell scripts and SUID programs are skipped.

---





## QUICK TIP

# Configuring network printers in Solaris 2.6

by Asim Zuberi

Configuring network printers in Solaris is simple and straightforward; all you need to know are the exact options of the commands. Not everyone has been using HP printers, which are very easy to configure with their `jetadmin` utility easily accessible from HP's Web site ([www.hp.com/cposupport/networking/software/ja245en.sol.html](http://www.hp.com/cposupport/networking/software/ja245en.sol.html)). There are lots of different kinds of printers out there that can be configured to a Solaris box via a network interface.

The printers I dealt with were QMS, Tektronix, and HP. The HP printers can easily be configured with the `jetadmin` utility in the `pkgadd` format. All you need to do is download and run `pkgadd` to install it on your computer. Change the directory where you've downloaded the software, and then follow the steps below:

```
# uncompress SOLd515.PKG.Z
# pkgadd -d SOLd515.PKG all
```

By default, the `jetadmin` software gets installed in the `/opt` directory.

```
# cd /opt/hpnp
# ./jetadmin
```

The `jetadmin` utility will display a menu on the screen.

- Choose option # 1, for the configuration. Another menu will be displayed.
- Choose option # 3 to add a printer to the local spooler. It will then prompt you to enter the IP address of the printer.
- Enter the network printer name or IP address.
- Once the IP address is entered, the `jetadmin` utility will ping to the printer to check the network connection. Once it receives packets back, it will display a list of suggested parameter values for the queue.
- Choose option # 3, for the Queue Class.

- Enter the class name, and then choose 0 to configure. Once it's configured, exit out of the `jetadmin` utility.

Use the `lpstat -t` command to check the status of the printer. If you'd like to make it the default printer, use the following command:

```
# lpadmin -d <class_name>
```

Now, let's talk about other network printers. Solaris comes with the `lpadmin` utility, which can be used to configure the printers. Here's how we'll use it:

```
#lpadmin -p <printername> -o \
  protocol=bsd,dest=<printdest> \
-T PS -I postscript -v /dev/null \
-I /usr/lib/lp/model/netstandard
```

where `printername` is whatever you want to call it and `printerdest` and the printer's IP address. We now need to change the security for our device:

```
# chmod 666 /dev/null
# chown root:sys /dev/null
```

You can then install the filters as follows:

```
# cd /etc/lp/fd
# for filter in *.fd ; do
  ↪name=`basename $filter .fd`
  ↪lpfilter -f $name -F $filter
  ↪done
```

```
# accept <printername>
# enable <printername>
```


Printers that can't be configured through BSD can be configured through TCP protocols with a specific port number instead.

```
# lpadmin -p <printername> -o \
  protocol=tcp,dest=IP:9100 \
-T PS -I postscript -v /dev/null \
-I /usr/lib/lp/model/netstandard
```



IP is the IP address of your printer and 9100 is the port number. Don't forget to install the filters for the printers as described above, and then accept/enable steps. Some other useful commands for the monitoring and management of print queues are

```
# lp -d <printer_name> (for the non default
printer)
# lp -n <file_name> (for multiple copies)
# lpadmin -d <printer_name/class_name> (to
make the printers as default)
# lpadmin -x <printer_name/class_name> (to
remove the printers)
# lpstat -t (shows the status of the
printers)
# lpstat -o (shows the jobs in the queue)
# cancel (to cancel the job)
```

Hopefully this will get you started with your network printer configuration. You can also consult Sun's Answerbook for further information. 

## 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](mailto:aja@missiondata.com).

**Werner Klauser** is an independent UNIX consultant working near Zurich, Switzerland. While not paragliding, enjoying his girls, or roarin' around on his Harley chopper, he can be reached via email at [klauser@klauser.ch](mailto:klauser@klauser.ch) or on his Web page at [www.klauser.ch](http://www.klauser.ch).

**Boris Loza** holds a Ph.D. in computer science from Russia. He used to work as a UNIX administrator and developer for 10 years. Currently he is working for Fidelity Investments Canada in the position of Data Security and Capacity Planner, doing IT security for UNIX, Windows NT, and Novell. He has a daughter, Anna, and likes reading computer and mystery books and watching movies. He can be reached at [Boris.Loza@FMR.com](mailto:Boris.Loza@FMR.com).

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

**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](mailto:robt@cymru.com), or visit his Web site at [www.cymru.com/~robt](http://www.cymru.com/~robt).

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

# 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](mailto:zdjcr@zd.com).

## Editorial

Editor ..... Garrett Suhn  
Assistant Editor ..... Jill Suhn  
Copy Editors ..... Rachel Krayer  
Christy Flanders  
Taryn Chase

Contributing Editors ..... Alvin Alexander  
Werner Klauser  
Boris Loza  
Paul Watters  
Robert Owen Thomas  
Asim Zuberi  
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](mailto:sun@zdjournals.com).

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

## Postmaster

Periodicals postage paid in Louisville, KY.

Postmaster: Send address changes to

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

## Copyright

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

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

Printed in the USA.

## Price

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

Our Canadian GST # is: R140496720.

## 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.

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

To see a list of our products, visit our Web site at [www.zdjournals.com](http://www.zdjournals.com).

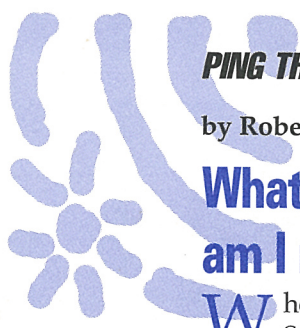


PERIODICALS MAIL

Sun Technical Support  
(800) 786-7638

C:7661905 00002096 04/00  
 ██████████  
 CLINTON TOWNSHIP, MI 48035-4218  
 ██████████

Please include account number from label with any correspondence.



**PING THE SOLARIS DUDE: SOLARIS Q & A**

by Robert Owen Thomas

**What version of BIND am I running?**

When taking over the administration of a Solaris box, it's often difficult to determine what modifications the previous administrator may have made. One such modification might be an upgrade of the name server daemon. Starting with Solaris 7, BIND 8.1.2 ships with all Solaris OS media. So it's a good idea to ascertain the version of BIND running on your nameserver.

To do this, there is a simple trick that can be executed remotely using the `nslookup(1M)` command:

```
: pudge; nslookup
Default Server: orc.research.cymru.com
Address: 192.168.0.254

> server goblin
Default server: goblin.research.cymru.com
Address: 192.168.0.4

> set class=chaos
> set type=txt
> version.bind
Server: goblin.research.cymru.com
Address: 192.168.0.4

VERSION.BIND      text = "8.1.2"
```

And there you have it! The host `goblin` is running version 8.1.2 of BIND. If the server returns the message:

```
*** goblin.research.cymru.com can't find
version.bind: Server failed ***
```

the version of BIND is release 4.9.5 or before. It also indicates that an upgrade of the BIND software is a must!

**What's a segmentation violation?**

A *segmentation violation* occurs when a pointer causes a memory reference to a segment that isn't in your address space. The kernel catches this attempt, and delivers a signal 11, or SIGSEGV, to your process. The default signal handling action is for the process to dump core and abort. The result, for you, is the abrupt end of your application and a (perhaps large) core file in a directory.

In general, this translates to the de-referencing of a bad pointer. The pointer may have been cleared already (through the `free()` call). The pointer may not have been assigned a value. The pointer may have been munged by a buffer overrun. Whatever the cause, the de-referencing of the pointer in question leads to a segmentation violation.

As an aside, a segmentation violation isn't the same as a bus error. A bus error results in signal 10 (SIGBUS) being delivered to the process. This is usually the result of memory mis-alignment. 