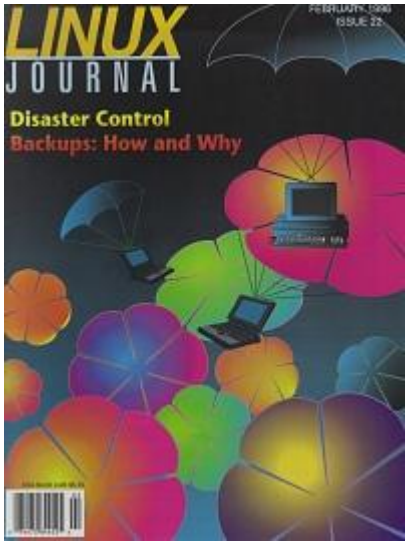
Linux Journal Issue #22/February 1996



Features

Backup Strategy *by Malcolm Murphy*
Malcolm tells us which files to backup and how often.
Tar and Taper for Linux *by Yusuf Nagree*
Learn to use tar and the friendly taper archival tools.
Backing Up in Linux *by Yusuf Nagree*
Prep yourself and your system for tape drive backups.

News and Articles

How to Read 950 E-mail Messages Before Lunch *by Jay D Allen*
A discussion of the use of e-mail filters on UNIX computers that use Sendmail-like systems.
X Forms: Review and Tutorial *by Karel Kubat*
Karel Kubat explores XForms, a graphical user interface toolkit for X.
Trade Show: Open Systems World/FedUNIX 1995
Trade Show: Comdex 1995

Columns

Letters to the Editor
Stop the Presses
Linux System Administration Maximizing System Security: Part 2
Product Review Caldera Desktop Preview II
Book Review Seamless Object-Oriented Software Architecture
New Products

Backup Strategy

Malcolm Murphy

Issue #22, February 1996

Everyone tells you how important it is to make backups. Explicit guidelines, however, are often lacking. Which files should you back up, and how often? This article will help you answer those questions, and use the answers to develop your own backup strategy.

Broadly speaking, we can identify two types of backup; the *system* backup, which is a backup of the operating system and applications (the things only the sysadmin can alter), and the *user* backup, which is a backup of the users' files (I don't know if anyone else uses these terms, but they'll do for the purposes of this article). As we shall see, system backups and user backups should be treated differently.

System backups

The reason for making system backups is to minimize the effort required, following a crash, to get the system up and running as it was before disaster struck. However, you don't want to spend half your life backing up your disk; no one said it was fun! The key to backing up effectively is to back up only that which is absolutely necessary to enable speedy recovery when disaster strikes.

Think about it: most of your system is pretty stable—the contents of /usr/bin don't change that often, do they? To make things even easier, you probably have a rough copy of your system already; most people install Linux from a distribution of some sort, then make their own customizations. The original distribution is likely to be the starting point of a recovery for many of us.

Linux differs from most other operating systems in that the operating system and a large number of applications are typically installed in one go, whereas DOS-based systems and even Unix-based systems other than Linux tend to be installed in a more piece-wise fashion; first the operating system, then each application, one-by-one. For those systems, it makes sense to back up the

whole system; usually a lot of time and care has been invested in setting the system up in the first place. By contrast, installing or re-installing a basic Linux system (complete with applications) is usually a quick and painless affair.

Having just said that most of your system is pretty stable, let's consider what is likely to change. One way you will customize your system is by adding new programs (software that didn't come as part of your distribution). When installing new software, you should be strict with yourself, and keep any new programs separate from those on the distribution. The best place for them is in the /usr/local hierarchy. As its name suggests, /usr/local is designed to contain programs that are local to *your* system. The advantage in doing this is that you can easily see which programs you can restore from your distribution, and which programs you need to restore from elsewhere.

Another thing you are likely to change is the configuration files the standard programs use. The behaviour of many standard Linux utilities is controlled by simple text files, which you can edit to tailor your system to your requirements. Sometimes distributions will “invisibly” edit some of these text files for you, based on your responses to certain questions, but often you have to edit them yourself.

A lot of the important files live in the /etc directory:

- /etc/printcap—describes how to communicate with your printers
- /etc/fstab—describes what file-systems you have
- /etc/passwd—contains a list of all users, and their (encrypted) passwords
- /etc/inittab—tells init how to set the system up for a given run level
- /etc/XF86Config—describes the initial setup of XFree86

Depending on your system, there are likely to be many others as well. As you can see, the /etc directory is very important, and the files it contains are likely to be the result of hours of work. I don't know if I'm typical, but I spent a long time just getting XF86Config exactly how I want it. The thought of going through that again is enough to make me shudder. Of course, some programs will use files in other places, but most of the basic Linux system is configured using files in /etc.

When you modify the configuration files used by an existing program, you can't move them somewhere else; the program (usually) looks for them in a particular place. Therefore, it is important to keep track of what changes you've made, so that, should disaster strike, you can get them back easily. Make a note of all the modifications you make to the system, no matter how trivial they seem at the time.

The best tool for the job is a pen and some paper. Write yourself long descriptions of what you've done, and why. Don't fall into the trap of thinking that in six months time you'll remember just how you got application Y to compile, or what the printcap entry to filter your postscript files through ghostscript was, because the chances are you won't. Even if you are installing new software in a separate directory so it's easy to keep track of, it won't hurt to write down what you installed, when you installed it, and if there were any things that didn't seem obvious at the time.

Now that we've identified *what kind* of system files we need to back up, let's consider *how often*. Just after you've made a change is probably the most important time, but don't forget to keep a backup of how the system was before the latest change, just in case things do go wrong later because of your change. The point is that things only change when you change them, which probably isn't very often, and the frequency of your backups should reflect this.

User Backups

ser backups are different from system backups, in that a user's files are liable to change frequently. It will almost certainly be impossible for you to have up-to-the-minute backups of a given user's file space, and you shouldn't even try. In backing up user files, you are offering your users a virtual safety net—reasonably recent copies of their files they can fall back on if they do something silly (like **rm * bak** instead of **rm *.bak**—it does happen!), or if the hard disk fails.

User backups will have to be done much more frequently than system backups, perhaps even daily (the cron program enables you to run programs at regular intervals, without having to issue the same commands each time—see the cron sidebar).

One useful feature of many backup programs (including tar) is the ability to backup only files which have changed after a certain date (the last time you did a backup, for example). This can drastically reduce the amount of work in a user backup, since a user is likely to be working on only a small number of files at a given time. You can combine full backups of your user space every so often with more frequent incremental backups.

While it is possible to use floppy disks for your backups, each disk can only hold a small amount of data. Many programs allow a backup to span several disks, but this means that you have to be there to change them while the backup is taking place. If you only have a small system with few users, then this might be feasible, but often it isn't. Magnetic or digital tapes are probably a better choice, simply because of their higher capacity. Linux supports a wide range of tape

drives, either via the ftape module or its SCSI support (digital drives are almost always SCSI). The price of tape drives has fallen quite dramatically in the last 18 months or so, and they are now a realistic option for many of us. Alternatively, your Linux box might be on the same network as another machine with a tape drive. Linux can access tapes on remote machines, but that is beyond the scope of this article.

Whatever media you choose, you should look after it. Your backup is there for when things go wrong, so it is important that you can rely on it. You should always verify your backups; it is often said that an unverified backup is worse than no backup at all.

You should also keep more than one set of backups. A popular strategy is based on the “grandfather-father-son” idea. You have three sets of backups; the last one (the son), the one before that (the father), and the one before that (the grandfather). When you do your next backup, you copy over the grandfather, so the son becomes the father, the father becomes the grandfather, and the grandfather is replaced with a new son. The advantage of this strategy is that should one of the sets fail, you at least have something to fall back on, but you don't have to make more than one backup at a time.

The next piece of advice might sound strange at first: always keep at least one backup well away from your machine, preferably in a completely different building. Why? Well, what if the building burns down? You can replace the machine, and get a new Linux distribution, but you won't be able to replace your backup tapes. The data on your computer is its most valuable and irreplaceable component, so treat it with care.

How?

Okay, enough of the chat—let's see some examples. There are many different backup programs available, both freeware and commercial. Each has its merits, but for these examples, I'm going to use tar (GNU version 1.11.2).

Suppose you've just installed a lot of new software in /usr/local, and think it's time you updated your backup of the whole /usr/local tree. You don't have a tape drive, so you're using floppies. A command like:

```
$ tar -cWMf /dev/fd0 /usr/local
```

will do the trick. The **c** option means create an archive, **W** means attempt to verify the archive after writing, **M** tells tar to span more than one floppy if it needs to, and the **f** option tells tar where to write the archive, in this case to /dev/fd0—the floppy disk drive. On many systems, you will have to be root in order to access /dev/fd0 directly.

Even though I've requested verify, it doesn't hurt to check. The command:

```
$ tar -tMf /dev/fd0
```

will show a list of all the files backed up. Depending on the size of your /usr/local tree, you might need several floppies. You could reduce the number of disks needed by using tar's compression option; the **z** flag will tell tar to filter the archive through gzip, thus saving disk space. A good idea? Well, yes and no. While it is attractive to save disk (or tape) space, compressing a lot of files together is risky. It means that the slightest corruption is likely to destroy the whole backup, whereas if the archive is uncompressed, it might be possible to read past any errors, and retrieve at least some of your data. Some programs compress files individually before backing them up, and this is probably a better idea.

I mentioned earlier that it is possible to back up files that have been modified since a certain time. With tar, you can achieve this using the **N** option. For example,

```
$ tar -cf /dev/ftape -N yesterday /home
```

will backup all files under /home which have been altered since yesterday, this time to a floppy tape device /dev/ftape. An alternative approach would be to use a combination of find and tar:

```
$ find /home -cnewer /etc/last_backup \
  -type f i-print > back_these_up
$ tar -cf /dev/ftape -T back_these_up
$ touch /etc/last_backup
```

Here, the find command finds all files under /home which have had their contents altered since the file /etc/last_backup was last modified, and writes their names to a file called back_these_up. The **T** option tells the tar command to back up the files listed in back_these_up. Then we **touch** the file /etc/last_backup, so that the next time we do this sequence of commands, we get the files that have been modified since this backup. Combining several commands like this is quite useful; as a side effect, we have a list of files that have been backed up, as well as the time of the last backup (the timestamp of the file /etc/last_backup).

Another thing we could do is to filter the list of files, so that certain files don't get backed up. For example, you might not want to back up object files, or DVI files, since they can easily be recreated from the source code (which is usually a much smaller file!). A simple **grep -v** will do the trick if there is only one kind of file you want to ignore; egrep can be used to ignore several kinds of files. Change the first line above to something like:

```
$ find /home -cnewer /etc/last_backup \
  -type f i-print | egrep -v '<<<>.o$|<<<>.dvi$' \
  > back_these_up
```

to ignore object and dvi files. It's also possible to do the same kind of thing with find for this simple example, although it doesn't have egrep's powerful regular expressions:

```
$ find /home -cnewer /etc/last_backup \
  -type f ! \( -name \*.o -o -name \*.dvi \) \
  -print > back_these_up
```

It is likely that your exact backup requirements can't be met easily by a single tar command, so don't be afraid to write your own little scripts to do the job. They can be as simple as the three line example above, or as complicated as you like. A few simple scripts, run regularly using cron, can make backing up a very easy process.

Backing up needn't be a protracted form of torture. It needs to be done, and as a sysadmin you have to do it, but a bit of planning and clear thinking goes a long way. It is easy to feel that you must have a complete current snapshot of your entire hard disk at all times, and equally easy to believe that a six-month old copy of a few files lying about somewhere will do. The best strategy lies somewhere in between.

Malcolm Murphy (Malcolm.Murphy@bristol.ac.uk) remembers a time when 256K of memory was considered more than enough for all your computing needs, instead of the bare minimum cache requirement, and wonders if we aren't just a little spoiled nowadays.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Tar and Taper for Linux

Yusuf Nagree

Issue #22, February 1996

In plain English, Yusuf Nagree explains archival backup with tar and his user-friendly taper program.

This article describes backing up files on a Linux system. Two programs are described—tar and taper. The first program is available from the Free Software Foundation under the GNU license and is included with most distributions of Linux. The second program is written by the author of this article and provides a more user friendly interface. It is also available under the GNU license and thus is freely available. Note that this article is not meant to be a full reference for either package, but merely an introduction to get you started. For full details, see the documentation that comes with each package.

tar

Nearly every form and clone of Unix (as well as other operating systems) comes with some version of tar. It is a standard program, and archives made on one machine should always be usable on other machines. The real problem with tar is that there is virtually no user-interface at all. All operations must be done via command line switches.

Making a Backup

tar can make backups to a hard disk file or to a tape drive as well as over a NFS link (which we won't cover here). The files to be backed up can be compressed using GNU gzip (or compress).

To make a backup, the basic form is:

```
$ tar [options] files_to_backup_or_restore
```

The most commonly used options are:

c

Creates a new archive.

z

Compresses the archive using GNU gzip.

Z

Compresses the archive using compress.

f *name*

Use ***name*** as the archive file or device. The default is documented as /dev/rmt0, although some people have changed this so that the default is /dev/nst0, /dev/tape, or even standard input. It is usually safer to explicitly give the device name of your tape drive all the time.

r

Append files to existing archive. Note that if you use ftape, this option will not work because of a limitation in the current ftape driver.

u

Append files to existing archive but only if they are newer than the files already in the archive. Once again, if you use ftape, this option will not work.

Thus, to create a compressed backup of your /etc directory in a file called etc_backup.tar, you would do:

$ tar czf etc_backup.tar /etc

Note that all subdirectories under /etc will be backed up as well.

If you now want to add the contents of /usr/local/etc, you would do: **$ tar rzf etc_backup.tar /usr/local/etc**

Suppose that you have now made some changes to the files, but not all of them. You can do:

$ tar uzf etc_backup.tar /etc /usr/local/etc

and tar will go through and append to the archive only those files that have been changed since the archive was originally created.

The above examples apply to backing up to a file on the hard disk. Backing up to a tape drive simply involves giving the filename of the tape device, usually /dev/ftape for floppy tape drives and /dev/st0 for SCSI tape drives.

Restoring

The two options that are relevant here are:

x
: Means e**x**tract file from archive. If no filenames are specified, all the files in the archive are extracted.

t
: Means print **t**able of contents; prints names of files that would be extracted but does not actually extract the files.

Thus, to restore the contents of the backup in the above example, you would do:

$ tar xzf etc_backup.tar

Note that tar does not put the files back where they came from, but rather creates a new tree based on the current directory. For example, if you were in the /usr/home/john directory when you issued the above command, you will find that a new subdirectory /usr/home/john/etc has been created and all the files are in that subdirectory. If you wish to restore the files whence they came:

```
$ cd /$ tar xzf etc_backup.tar
```

Note that doing this is **very** dangerous, since old files are over-written without warning. This can have dire consequences if not used properly. It is often much safer to restore in your home directory or /tmp and then copy the files to their correct location after you have checked that nothing horrible will happen.

To restore an individual file or directory, simply specify the name after all the tar arguments. For example, to restore just the hosts and the passwd file:

$ tar xzf etc_backup.tar etc/hosts etc/passwd

Note that the full pathname (excluding the leading **/**, which tar explicitly does not store) needs to be specified.

Restoring Distributions

Most people who write software for Linux and make it available to others distribute it via tar files. For example, say that you have downloaded a new game, best_game-1.3.tar.gz. To install (restore) this game in your home directory:

```
$ cd$ tar xzf best_game-1.3.tar.gz
```

The game and all its applicable files will then be restored. If the author of the program has followed normal convention, all the files will be in a directory called best_game-1.3. Note that when restoring files from an unknown source, it is a very good idea to restore the files in your home directory, examine the files, and then when sure everything is correct, move them to the location suggested by the author. This way, you will avoid inadvertent file overwrites. It is also best to first use the **t** option to see whether the author has put the files in a subdirectory. If not, make a subdirectory and use it:

```
$ cd$ mkdir worst_game-0.1$ cd worst_game-0.1$ tar xzf worst_game-0.1.tar.gz
```

Other Options

Some of the other options that tar supports are:

M

Tells tar to use multi-volume archives. If tar comes to the end of the floppy or tape, it will not abort with an error, but prompt for insertion of a new floppy or tape, which it calls a “volume”. Each volume contains a stand-alone archive file and you don't need all the volumes to extract files, but if a file is split across two volumes, you will need to extract that file with the **-xM** option. Note that some tape devices, such as DATs, do not work with this option.

N *DATE*

Tells tar to operate only on files that are newer than ***DATE***. Thus, you can tell tar to backup only files that are newer than a certain date. ***DATE*** is specified in the same format produced by the **date** command. Normally, directly before doing one backup, you use the **date** command to record the date and time when the backup was made, like this:**$ date > last_backup**Then the next time you are backing up, you can backup only files that have been changed during or since the last backup by including the option **-N "cat last_backup"** in tar's command line.

T *FILENAME*

Tells tar that a list of files to backup/restore is in ***FILENAME***. For example, **tar czf /dev/ftape LIST_FILES** would create an archive containing files that are named in the LIST_FILES file. The LIST_FILES file is simply a straight text file with one filename on each line.

v

verbose mode.

h

When tar comes across a link, it normally stores details about that link. If this option is given, tar will actually store the file pointed to by the link and pretend the link doesn't exist. Use this with caution since you can end up with many different copies of the same file.

W

Causes tar to verify the archive after it has written it. It will not work on tape drives that cannot rewind.

P

Normally tar strips the leading / from a pathname so that when you restore, the file is restored in a directory relative to the current one (see the above example with /usr/home/john). By specifying this option, the file is restored from where it was backed up. Use this with caution since you can inadvertently overwrite a file on your hard disk.

More details can be found in the tar man page as well as in the info files that come with the tar sources.

Using Floppy Drives

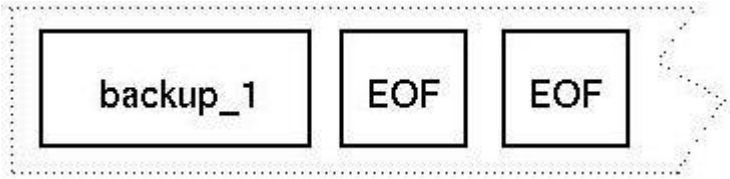
Although most backups are done on tape drives, it is sometimes useful to use your floppy drive (which can be very slow and frustrating!). To use your floppy, give tar the filename /dev/fd0 for the first floppy (drive A: in DOS parlance) and /dev/fd1 for the second floppy (drive B:). So, to make a backup of your /etc directory to “drive A:”

$ tar czf /dev/fd0 /etc

Note that the existing contents of the floppy will be totally overwritten. When using floppy drives, the **-M** option is very handy since after one floppy is full, tar will automatically prompt for the next floppy.

Multiple Archives on One Tape

If you use ftape, you cannot append files to an existing tar backup. Therefore, if you make a backup that occupies only 10 MB, you will have wasted the rest of the tape. Fortunately, there is a way of using the rest of the tape, using the mt program. After tar has written a backup to the tape, it writes two EOF marks on the tape. Schematically, your tape now looks like:



You can tell mt to advance to these EOF marks by

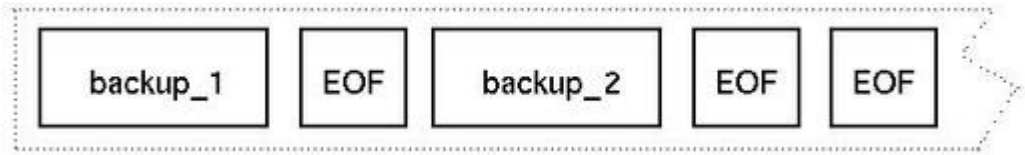
$ mt -f /dev/nftape fsf 1

Note that you must use the non-rewinding device (/dev/nftape), because if you don't, after mt has repositioned the tape, it will automatically be rewound and your repverbose mode positioning will be lost. The **fsf 1** is an mt command that advances the tape to the first EOF mark.

Your tape will now be positioned at the end of the first backup, and you can create another backup at this position:

$ tar czf /dev/ftape files_for_backup_2

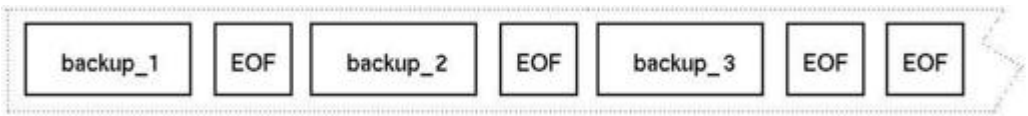
Because you used the rewinding device, the tape will rewind after the tar backup has been written. Your tape will look like this:



Should you wish to add a third backup, you can do the following:

```
$ mt -f /dev/nftape fsf 2<
$ tar czf /dev/ftape files_for_backup_3
```

The **fsf 2** advances past two EOF marks—hence, past the first two backups. Your tape will look like:



and will be positioned at the beginning of the tape because you used the the rewinding device to do the backup.

If you wish to restore files from the first backup, you can do the following:

```
$ mt -f /dev/ftape rewind<
$ tar xzf /dev/ftape files_from_backup_1
```

This first line ensures that the tape has been rewound. The second line then restores files from the first backup. If you wish to get files from the second backup, do the following:

```
$ mt -f /dev/ftape rewind
$ mt -f /dev/nftape fsf 1
$ tar xzf /dev/ftape file_from_backup_2
```

Similarly, to restore files from the third backup:

```
$ mt -f /dev/ftape rewind
$ mt -f /dev/nftape fsf 2
$ tar xzf /dev/ftape files_from_backup_3
```

You can continue adding backups until your tape is full. Note that strictly speaking, you do not need to rewind the tape each time—if you are sure that the tape is rewound, you can skip the rewind command; however, I would recommend that you always rewind prior to use to avoid problems. If you are already at the beginning of the tape, the rewind command will not take any time at all. You are likely to get into all sorts of bother if you assume that the tape is rewound when it is not. You can find out where your tape actually is by:

$ mt -f /dev/nftape status

As you can imagine, maintaining backups this way is awkward and very time-consuming since if you do not know what is on the third backup, you have to rewind your tape, advance to the third backup and then read what is there.

Another problem with tar is that, when using the compression mode, tar compresses all the files and then writes them to the tape. This leads to a very serious problem. If your tape somehow becomes corrupted, you lose all the files after the corruption occurred.

Because of the above, and because of tar's command-line user-interface, the author was tempted to fill the gap with a user-friendly backup program—hence, taper was born...

taper

Although tar is a very powerful and flexible program, it has no friendly user interface. taper provides many of the same features as tar, but provides a nice user-friendly interface.

See below for locations of the taper sources. At the time this article was written, the current version was 5.4. taper requires a recent version (1.9.6 or greater) of ncurses that supports “forms”, which can be retrieved from any GNU mirror site or from the primary GNU ftp site, prep.ai.mit.edu. It is easy to configure, build, and install for Linux; installation instructions are included in the INSTALL file. As of this writing, the current version of ncurses is 1.9.7a and can be found in the file ncurses-1.9.7a.tar.gz.

To build and install the latest ncurses, I did the following as root:

```
$ cd /usr/local/sr
$ tar xzf ncurses-1.9.7a.tar.gz
$ cd ncurses-1.9.7a
$ ./configure --with-normal --with-shared
```

```
    --with—debug --disable-termcap
$ make
$ make install
```

The steps you need to take may be slightly different; the INSTALL file gives plenty of details. The **--with-shared** option is especially important if you have ELF libraries and don't remove previous versions—otherwise, compiling taper may not work.

To make a binary of taper, issue the following commands:

```
$ tar xzf taper-xx.xx.tar.gz
```

where ***xx.xx*** is the the version that you have obtained. Next, you may need to edit the Makefile (e.g., if you are using a SCSI drive, remove the [cw]#[ecw] that is in front of the line **-DHAVE_SCSI**, or if you are using the new zftape driver, remove the **#** that is in front of the line **-DUSING_ZFTAPE**—see the Makefile for more details). Then type:

```
$ make clean
$ make all
$ make install
```

By default, the programs will have been installed in /sbin and the manual page in /usr/man/cat1. You can change these if you wish by appropriately editing the Makefile. If you do not have write permission to these directories, the files will remain in the current directory and will not be copied across.

taper Basics

Some terminology first:

archive
: This describes all the files on a tape (or hard disk file). You can restore files from an archive or add files to an archive.

volume
: Each archive is divided into one or more volumes. Each time you back up a set of files to an archive, a new volume is created. For example, if in one session you back up /etc, then the archive contains one volume. If you then add /usr/local/etc to the archive, another volume is created so that you still have one archive, but two volumes. Note that if you backup /etc and /usr/local/etc in one backup session, only one volume is created.

preferences
: It is possible to customize various aspects of taper—everything from screen colours, to how taper behaves when it encounters soft links. Each customizable option is called a preference.

file set

It is possible to store commonly selected groups of files into a file set. For example, you may periodically want to backup only /etc, /usr/home/user1 and /usr/local/etc. Rather than having to explicitly select those directories every time you wish to make a backup, you can select them once and then save them to a file set. Subsequently, when you are making backups, you need only load in the file set to automatically select those directories.

incremental backup

taper supports two modes of backup, full backup and incremental backup. With full backup mode, all the files and directories you select are backed up. In incremental mode, taper is intelligent. If the file you have selected for backup already exists on an archive, taper will back it up only if the file has changed. If the file hasn't changed, it won't be backed up. This makes backing up large directories very quick and easy since only the changed files are backed up.

most recent restore

It is possible that you will have many copies of the same file on an archive —from old versions to the most recent version. taper can automatically detect which is the most recent and restore only that; you need not manually determine and select it.

When you create an archive, taper stores all the information about files on that archive (such as filename, file size, backup time, etc.) into a file called the *archive information file*. This file is stored on the hard disk in a directory (default is ~/.taper_info). In future, when accessing this archive, taper uses the archive information file to quickly gain access to all details about the archive— this speeds up performance since the tape doesn't need to be accessed.

The downside of this is that you need to make sure that this information file does not get deleted or corrupted (don't despair if it does, though, since you can recreate it). Also, if you wish to restore files on a different machine, you have to make sure that you either reconstruct the info file on the new machine, or take a copy of the info file with you on a floppy (or via an ftp transfer).

Each archive created is allocated a unique archive **ID** and you use that in future for accessing the archive if you don't have the tape handy.

You need to tell taper the name of the backup devices (both rewinding and non-rewinding). You can do that by giving the command line options **-f** (or **--rewinding-device**) for the rewinding device and **-n** (or **--non-rewinding-device**) for the non- rewinding device. Alternatively, you can start taper and then specify the names using the **Global preferences** option.

If you compiled with **-DHAVE_SCSI** option on, the default names are /dev/st0 (rewinding) and /dev/nst0 (non-rewinding). If you compiled with **-DUSING_ZFTAPE**, the default names are /dev/qftape(rewinding) and /dev/nqftape (non-rewinding). Otherwise, the default names are /dev/ftape (rewinding) and /dev/nftape (non-rewinding). It is also possible to set default names using environment variables—**TAPE** is the name of the rewinding device and **NTAPE** is the name of the non-rewinding device. Alternatively, you can use a preference file to set defaults (see below).

Making a Backup

Start taper, by typing:

$ taper

You will then be presented with the main taper window. There are three main modules—backup, restore and mkinfo, as well as preference management options. Select the backup option.

If an archive exists, you will be asked whether you wish to append files to it, or whether you wish to overwrite it. As with all dialog boxes, the space-bar toggles between the options, and **ENTER** will select the currently highlighted option.

If the archive doesn't exist, you will be prompted for the archive title.

Next you will be prompted for the volume title.

You will then be presented with a screen with three panels. The top left shows the current directory on the hard disk, the top right shows what's currently on the archive and the bottom panel is used to show which files have been selected for backup. At the top of the screen is the archive ID and archive title.

To move between panels, press the **TAB** key. To get help on keys, press **H**.

You now need to select which files and directories you wish to backup. Use the cursor keys to move around the directory. Pressing **ENTER** when the highlight is on a directory will move into that directory.

When you find a file or directory you wish to back up, press **S**. The file/directory will then be sized and moved to the bottom window—if you selected a directory, taper will check with you that you really want to back it up. Press **ENTER** to confirm. To disable the confirmation, change this in global preferences (Prompt Directories).

In the bottom window, the file/directory will be printed as well as its size. Also, to the left of this, there will be an **I** or **F**. This indicates that the file/directory will be backed up in incremental mode (**IM.**) or full backup mode (**F**). To toggle between **F** and **I** modes, press **S** when the highlight is on the selected file or directory.

When you select directories, all directories under that directory are recursively included.

If you wish to deselect a file, move the cursor to the bottom window (using TAB) and then move the highlight to the file/directory you wish to deselect. Press D and the file/directory will be deselected.

If you select a file (e.g., /usr/home/john/xyz) and then select the directory in which the file resides (e.g., /usr/home/john), taper automatically recognizes that the file has been selected twice and will put brackets around the file (/usr/home/john/xyz) to tell you so. When doing the backup, the file will be backed up only once.

When you have finished selecting, press F and taper will commence the backup. Pressing Q at any time will abort the backup.

Restoring Files

Select restore from the taper main menu.

You will then be presented with a list of all the archives taper knows about. They are sorted in archive ID order and the archive title is also printed. The highlight will be on the archive that is currently in the tape drive. Move the highlight onto the archive that you wish to restore from and press ENTER.

You will then be presented with three panels. The top left shows the files and directories currently on the archive, the top right shows a summary of the whole archive and the bottom panel is used to show the directories and files selected for restoring.

Use the cursor keys to move the highlight to select which files you wish to restore—pressing S selects the file/directory the highlight is currently on. Directories are recursively selected.

When you have selected a file/directory, it is transferred to the bottom window. In a similar way to backup, restore will put brackets around files selected twice.

In the select window, after the filename, the volume number is printed. This will show either a volume number or **M**. If M appears, taper is operating in most

recent restore mode and will restore only the most recent copy of that file. If a volume number is displayed, the file will be restored from that volume, regardless of how recent the file in that volume is. You can toggle between the two modes by pressing **S** in the select window.

To deselect a file, position the highlight on the file you wish to deselect and press **D**.

When you have finished selecting files for restore, press F and taper will commence the restore. Pressing Q will abort the restore operation.

Mkinfo

If you lose the archive information file (it gets deleted, corrupted or you try to restore on another machine and forget to take the archive information file with you), you can reconstruct the info file. Simply put the tape in the drive and select mkinfo from the taper main menu.

Preferences

Below is a list of the more common preferences. For a complete list, see the man page.

compress
: Tells taper whether to compress files it writes to an archive. Default is TRUE.

log file
: Where taper logs activity to. Default is ~/.taper_log.

Klog level
: The level of logging from 0 (no logging) to 3 (verbose). Default is 2.

prompt directories
: Whether taper confirms before selecting directories in restore and backup. Default is FALSE.

incremental backup
: Whether incremental backup is used as a default. Default is TRUE.

most recent restore
: Whether, by default, taper should restore the most recent file or the file the user specifies. Default is TRUE.

exclude compress

Certain types of files can bypass the compression facility—e.g., by default, taper doesn't try to compress .gz or .gif files. This preference specifies which files not to try and compress. The preference is simply a string with the files you wish to exclude given as a space-separated list (e.g., default is **.gz .gif .Z**)

exclude files
: Certain types of files can be excluded from the archive, even if explicitly specified—e.g., by default, taper doesn't try to back up .o files. This preference specifies which files to automatically exclude. The format is the same as the “exclude compress” preference and the default is .o ~

You can save your preferences to customize your particular setup. There are two ways to s: one is to a preference file and the second is to a command line file which can then be used to start taper in the future. Simply select the appropriate option from the main menu.

taper looks for a preference file in the following order:

- The filename given by the -p (--preference-file) option on the command line
- The filename given by the environment variable TAPER_PREFS
- The file ~/.taper_prefs
- The file /usr/local/etc/.taper_prefs
- Internal defaults

Leading Zeros

Some tape drives write zeros to the beginning of a tape and this can cause confusion with taper, which thinks it has reached the end of the tape when it detects zeros. To find out if your tape drive does this, put a tape in the drive and run the testzero program—note that the tape will be overwritten. taper will test your tape and print the result on the screen.

If taper says that your drive writes leading zeros, you will have to run mktape on every tape before you can use it with taper.

Erasing Tapes

People using floppy tape drives have to both format and erase tapes. These users must either format tapes using DOS, OS/2 or WINDOWS or buy pre-formatted tapes. Erasing tapes is done automatically by taper.

People with SCSI tape drives generally don't need to format tapes. Some SCSI tape drives don't even need erasing (e.g., DAT). To tell taper not to erase tapes before using them, change the **erase tape** option in the backup preferences menu and save the preferences. **Run mktape on all tapes before you use them so that taper doesn't think they are bad.**

If you have a SCSI drive and are not sure whether you need to erase tapes, tell taper not to erase tapes and see what happens.

/proc File-System

Linux has support for a /proc file system. This is a directory that looks like a normal directory, but actually contains information about the current machine state. It is useful for programs like **ps** which can read this directory and print the information contained in it.

However, we obviously don't want to back up this directory. You can tell taper to automatically exclude this directory. Run the which_device program:

$ ./which_device /proc

and the device on which the /proc directory is mounted is printed. Most probably this is 1, in which case you don't need to do anything because this is taper's default. If it is not 1, tell taper this via the backup options or via the command line (**--proc-device num**).

Memory Considerations

If you have many files to backup, taper can end up using quite a bit of memory. If you wish to minimize the amount of memory taper uses while running, edit your Makefile and un-comment the line (i.e., remove the **#**] in front of the line):

DEFINES=-DMEMORY_TIGHT

Now, when taper runs, it will be quite memory efficient. The downside (of course, there's a downside!) will be that performance will not be as good. However, on most machines, you will not notice a performance degredation until you reach 2,000-3,000 files.

SCSI Drives

There is a special preference that applies only to SCSI drives. This is the **--block-size** option. The SCSI kernel tape driver expects data to be presented to it in blocks of a maximum size—the default is 32K. For this reason, taper writes data in blocks of 28K by default. However, should you wish to change that, you can

do so with the [cw]--block-size[ecw] optioni—for example, some tape drives may function more optimally if data is written in blocks of, say, 64K. Note that this *must* be less than the SCSI kernel tape driver's maximum size.

You can change this option for non-SCSI drives, but it won't really affect performance.

File Sets

There is no distinction between file sets made in restore and file sets made in backup—they can be used interchangeably.

To make a file set, enter backup or restore and select the files and directories you wish to designate as your file set. Then press B and taper will prompt you for a name to give to the file set. After you enter the name, the file set will be saved.

Next time you wish to backup this particular file set, press L in backup or restore, and taper will show you a list of the file sets it knows about. Select one using the arrow keys and ENTER. This particular file set will then be loaded.

Conclusion

taper was designed to make backing up your Linux file system easy and painless. The traditional Unix utilities, tar and cpio, are very powerful, but they are not very user friendly. With Linux becoming more popular with non-hackers, another backup solution was badly needed. I hope taper fills this gap.

There are times, however, when you should use tar rather than taper. They are:

- When you will be doing backups on one UN*X system and restores on another—e.g., you make a backup on your Linux system and you restore on a Xenix system. As yet, taper has not addressed cross-platform archive compatibility—it may work, but it is not guaranteed. If you do wish to use taper to do this, test it thoroughly first.
- If you need to do remote file accessing—e.g., need to access files on host:/directory. taper does not support this yet, and it may be a while before it is added.
- Software developers distributing their programs as source files are still better off using tar because to distribute as taper files means also having to distribute the *archive information file*, which the end-user would have to place in the ~/.taper_info directory—another step confusing to novices.

Unless you are in one of those situations, taper should be adequate for most of your needs. It is certainly easier to use than tar and cpio.

As this product is under development, suggestions, bug-fixes, comments, etc. are all welcome. Similarly, short messages saying that taper works for your system are greatly appreciated since it gives me an idea of how many people are using taper and what sort of hardware it works on. This can allow me to help other people who have similar hardware.

Yusuf Nagree is a part time doctor and a full time Linux hacker (aargh—sorry, full time doctor and part time Linux hacker). He has been a computer buff since his dad bought him a ZX-80 in 1980 and has had various computers over the years. Bored with DOS, OS/2 and Windows, the aspect of Linux he finds most enjoyable is the community spirit and general willingness to help and share knowledge and experience.

Archive Index Issue Table of Contents

Advanced search

Backing Up In Linux

Yusuf Nagree

Issue #22, February 1996

Want to start backing up your Linux system properly, but don't know what's available to do the job? Yusuf Nagree helps you configure and use available backup hardware.

There are currently three main types of tape drives available for the PC (and hence Linux). There are those based on SCSI interfaces, those based on the floppy drive interface (QIC-40, QIC-80, QIC-117, QIC-3010 and QIC-3020), and QIC-02 type drives. A new breed of tape drives that attach to the IDE interface is coming, but at the moment, Linux support is minimal. There is no Linux support for drives that attach to the parallel port.

You cannot simply attach a tape drive to your machine and expect Linux to automatically recognize it. You need to tell the kernel about it, and this can involve recompiling the kernel. In addition, you need entries in your /dev directory so that programs under Linux can access the tape drives.

Below, I will describe how to set up the two most common types of tape drives.

Floppy Interface Tape Drives

Tape drives that connect to the floppy drive interface have the advantage that no separate interface card is required; they are, therefore, fairly cheap and reliable. Thus, this breed of tape drive has been very popular.

One of the first tape drives of this type was the Colorado Jumbo 250. The tape drive cable for these drives attaches to the floppy drive interface. The QIC-80 specification defines how to access these tape drives. Many other tape drives, including the Iomega 250, Conner C250MQ, Wangtek 304, and Colorado Jumbo 350 are QIC-80 compatible.

Tape drives that use the QIC-80 specification require a program (actually, a "Loadable Kernel Module") called *ftape*, written by Bas Laarhoven and Kai Harrekilde-Petersen, which at the time of writing is at version 2.03b. QIC-117 and QIC-40 formats are also supported by ftape, as is QIC-3010 and QIC-3020 (i.e. QIC-WIDE) in an experimental form. A complete list of tape drives supported by ftape can be found in the vndors.h file in the ftape source distribution. Support for some enhanced controller boards is also provided—specifically Iomega Tape Accelerator, Colorado FC-10, and Mountain MACH-2. Support for the FC-15 and FC-20 high speed Colorado controller boards is not yet provided.

Most Linux distributions include ftape; if yours doesn't, you will have to download it (see *Tape Resources* sidebar). Unless you are using module version support in your kernel (if you don't understand what this means, you can assume that you are not using it), ftape will need to be recompiled each time you update your kernel.

Compiling ftape

First, go to your source directory and unpack the sources:

```
:$ cd /usr/local/src
$ tar xzf ftape-2.03b.tar.gz
```

You will end up with a directory ftape-2.03b; all the ftape source files will be in this directory along with some documentation files. You now need to compile ftape to end up with a file ftape.o:

```
$ cd ftape-2.03b
$ make clean
$ make dep
$ make all
```

Next, you need to make sure that your kernel has been compiled with ftape support built in. Recompiling the kernel is beyond the scope of this article (see the Kernel-HOWTO more details), but basically, do:

```
$ cd /usr/src/linux
$ make config
```

to accept all the default values for all options (unless, of course, you *do* want to change them), and when you get to the QIC-117 option, answer **Y**. Leave the **NR_FTAPE_BUFFERS** at the default value of 3. Then recompile the kernel (usually **make clean dep; make zImage**) and install the new kernel. Don't forget to re-run lilo if you use it.

If you are using very recent kernels (1.3.30 and above), you will have to use **zftape**. zftape (written by Claus Heine) is based on ftape but provides support

for the dynamically loaded buffers provided by the later kernels. You compile it exactly the same way as ftape. Installing it is also done the same way as with ftape, except that you will use the name zftape instead of ftape. When configuring your kernel, you will not get asked any questions about QIC-117 options or **NR_FTAPE_BUFFERS**.

Even if you are not using recent kernels, zftape provides some very good enhancements over the basic ftape package, including software compression, and it is well worthwhile upgrading to get it.

Making The Devices

Next, you need to make sure that the /dev entries have been created correctly for ftape. Once again, if you have a Linux distribution, this will more than likely have been done, otherwise; you will have to create them manually. Do:

```
$ ls /dev/*rft* /dev/*tape*
```

and you should have at least the following files:

```
/dev/rft0
/dev/nrft0
/dev/ftape
/dev/nftape
```

If you do not, create them (you will have to be root):

```
$ mknod -m 666 /dev/rft0 c 27 0
$ mknod -m 666 /dev/nrft0 c 27 4
$ ln -s /dev/rft0 /dev/ftape
$ ln -s /dev/nrft0 /dev/nftape
```

Alternatively, if you have a script called MAKEDEV in your /dev/ directory, you can simply run this to have your devices created correctly.

```
$ cd /dev
$ ./MAKEDEV ftape
```

If you are using the zftape package, there are certain other device names that you will require. They can be created from the zftape Makefile by:

```
$ make mknod
$ ln -s /dev/rft0 /dev/ftape
$ ln -s /dev/nrft0 /dev/nftape
$ ln -s /dev/qft0 /dev/qftape
$ ln -s /dev/nqft0 /dev/nqftape
```

Installing ftape

You next need to make sure that you have the module utilities. This set of utilities allows the ftape driver to be loaded so that the kernel can access it. You will need the program **insmod** which should be in /sbin. If you do not have it,

obtain the latest version (see **Tape Resources**), compile it (which is quite easy), and install it.

```
$ tar xzf modules-xx.xx.tar.gz
$ cd modules-xx.xx
$ make clean
$ make
$ make install
```

Note that if you are using modules-1.1.87, you must replace insmod.c and insmod.h with the ones that come in the ftape distribution. To avoid these problems, obtain a copy of the latest modules, which contains many other bug fixes as well.

Before you can use the tape drive, you must load the ftape program. You will have to be root to do so.

```
$ insmod ftape.o
```

This must be done every time you boot up Linux. If you do a lot of tape drive work, it is a good idea to include this in your rc.local startup script so that every time you boot up the tape driver is automatically loaded.

Tape Preparation

There are two steps to using a new tape with Linux. The tape must be low-level formatted. You can actually buy pre-formatted tapes (and they are only a couple of extra dollars and well worth the money), but if you have bought an unformatted tape, you will have to format it yourself. There is no Linux program available to format tapes, so this must be done under DOS, OS/2 or WINDOWS. DOS programs known to format tapes correctly include Norton Backup, Colorado Systems Backup Program (shipped with Jumbo drives), and Conner Backup Basics.

Next, the tape has to be prepared for use by ftape which has to write headers and sector maps. You can use **mt** do this preparation (which is known as erasing a tape).

```
$ mt -f /dev/ftape erase
```

mt comes as part of the cpio package from GNU. See below for locations.

If you are using zftape, your device name is /dev/qftape and hence, you need to issue:

```
$ mt -f /dev/qftape erase
```

Common Problems

- **PCI motherboards** Some PCI motherboards have problems using ftape. The difficulty seems to lie with the Guaranteed Access Timing (GAT) option in the BIOS. This option must be disabled for ftape to work correctly. Note that some of the later Intel boards (1.00.10AX1 and later) have this setting permanently disabled.
- **insmod says "wrong kernel version"** As mentioned above, ftape will only work with the kernel that was running when it was compiled. Each time you change kernel versions, you will need to recompile ftape.
- **Unable to use floppy disk** Because both the tape drive and the floppy disk use IRQ 6, it is impossible to concurrently use the tape drive and floppy disk drive. Therefore, if you try to use the floppy disk while ftape is installed, you will get an error message. Similarly, if you try to install ftape while you have a floppy disk mounted, you will get an error. This is a hardware limitation and has nothing to do with ftape.
- **ftape seems to hang after accessing tape** Earlier versions of ftape had problems with retry errors. Updating to the latest ftape (currently v2.03) usually solves these.
- **No such device error** When trying to use tar or some other program that accesses the tape drive, you get a "No such device error". This is because you have not installed the ftape driver using insmod as described above.
- **No such file or directory** When trying to use tar or some other program that accesses the tape drive, you get a No such file or directory error. This is because you do not have a /dev/ftape and /dev/rft0 entries in your /dev directories. Create them using MAKEDEV or using the method outlined above.

SCSI Tape Drives

SCSI tape drives can be difficult (or very easy) to get working but they are generally quicker, and more reliable, and there is no problem using floppies and tape drives simultaneously. The downside is that they are usually much more expensive than their floppy drive equivalents.

Compiling with SCSI Support

You have to ensure that the kernel you are running has support enabled for your SCSI adaptor. Change to the kernel directory and start the kernel configuration script:

```
$ cd /usr/src/linux
$ make config
```

Press **ENTER** for all the options to accept the default values, until you come to the question **CONFIG_SCSI**. Type **Y** for this option. Press **ENTER** until you come to the question **CONFIG_CHR_DEV** and type **Y** for this option. Continue pressing **ENTER** until you come to your SCSI adaptor and answer **Y** to this question. You then have to recompile the kernel, as mentioned above.

When you boot up, you should now get a message similar to this (the numbers, and details may, of course, vary):

```
Detected SCSI tape st0 at scsi0 id 4, lun 0
scsi: Detected 1 SCSI tape 1 SCSI disk total
```

Making Devices

Next, you need to make sure that the SCSI devices have been created in your /dev directory. Once again, if you have a Linux distribution, this will more than likely have been done. Otherwise, you will have to create them manually. Do:

```
$ ls /dev/*st*
```

and you should have at least the following files:

```
/dev/st0
/dev/nst0
```

If you do not, create them (you will have to be root):

```
$ mknod -m 666 /dev/st0 c 9 0
$ mknod -m 666 /dev/nst0 c 9 128
```

Note, the above assumes that you are using the *first* tape on the SCSI bus. If you have two tapes, and you want to use the second one, change the device names to /dev/st1 and /dev/nst1 and create them:

```
$ mknod -m 666 /dev/st1 c 9 1
$ mknod -m 666 /dev/nst1 c 9 129
```

If you have a more recent distribution, chances are you will have the MAKEDEV script available in your /dev directory. You can create all the appropriate devices by simply running that script:

```
$ cd /dev
$ ./MAKEDEV st0
```

Tape Preparation

Unlike floppy tape drive tapes, SCSI tapes generally do not need formatting. They may, however, need erasing for use under Linux. To erase a tape, do:

```
$ mt -f /dev/st0 erase
```

DAT tapes, however, do not need erasing. The easiest way to find out if a particular tape needs erasing is to try using it without erasing first. If you can, well and good; if not, you will have to erase prior to use.

Device Names

Accessing a tape drive is very similar to accessing a file on the hard disk, except that a tape drive has two filenames. For ftape, these two names are generally /dev/ftape and /dev/nftape. If you use zftape, the two device names are generally /dev/qftape and /dev/nqftape. For SCSI, the names are /dev/st0 and /dev/nst0 for the first SCSI tape device, /dev/st1 and /dev/nst1 for the second SCSI tape device, and so on.

When the tape drive is accessed by the first filename (/dev/ftape, /dev/qftape, or /dev/st0), we are said to be accessing the *rewinding* device. When the tape drive is closed, the tape is automatically rewound to the beginning. When accessing the tape drive via the second filename (/dev/nftape, /dev/nqftape, or /dev/nst0), we are using the *non-rewinding* device and when the tape drive is closed, the tape is left where it is.

Some applications need to use both devices and you will need to specify the correct names.

Testing Using tar

Now that you have set up your system for a tape drive, you will want to test it. GNU tar is the de facto backup standard under Linux and comes will all distributions. If you do not have it, obtain it from a site near you.

In the examples below, I will use ***dev_name*** to indicate your device name. As mentioned above, this will probably be /dev/ftape if you use ftape, /dev/qftape if you use zftape, and /dev/st0 if you use a SCSI drive.

Put a freshly prepared tape into the tape drive and try to make a small backup:

```
$ tar cf dev_name /etc
```

This should backup your /etc directory. You can now check to see if the backup was made correctly by:

```
$ tar df dev_name
```

Note that if you use ftape, you cannot use the **Ar** options to tar because of limitations in the current driver. That is, you cannot append files to an archive. You will have to use mt to move the tape to the end of one archive and then create another archive.

A more detailed look at tar is provided in the Tar and Taper for Linux article.

Please note that the above sites are **very** busy. In the interests of preserving your sanity and minimizing network traffic, find a mirror (and there are many) near you and use that. Both tsx-11 and sunsite will print a list of mirrors if you try to log on when they are busy.

Further Information

The HOW-TOs are an invaluable source of information. There is a HOW-TO for both ftape and for SCSI.

Yusuf Nagree is a part time doctor and a full time Linux hacker (aargh—sorry, full time doctor and part time Linux hacker). He has been a computer buff since his dad bought him a ZX-80 in 1980 and has had various computers over the years. Bored with DOS, OS/2 and Windows, the aspect of Linux he finds most enjoyable is the community spirit and general willingness to help and share knowledge and experience.

Archive Index Issue Table of Contents

Advanced search

How To Read 950 E-mail Messages Before Lunch

Jay D. Allen

Issue #22, February 1996

A discussion of the use of e-mail filters on Unix computers that use Sendmail-like mail systems.

I have heard it said that electronic mail is the biggest reason that people first go to the Internet, and that Usenet news is why they stay. My question is, what is the single biggest reason that people leave the Internet? The answer is junk mail! Well, maybe they haven't left. It just looks that way because they **never** answer e-mail. It's easy to understand why in a public forum that spans the globe and includes tens of millions of members, people might be receiving more mail than they can read.

There are thousands of opportunities to subscribe to mailing lists, and interneters can quickly overfill their plates. I know, I've done it. In this article I'll discuss a powerful set of tools that allow you to get control of your in-box and reduce your chances of heart trouble related to starting your mail reader. These tools are called “E-mail filters”. E-mail filters are programs that sort mail based on a your directions. For instance; all mail from my brother should be moved from my in-box mail folder to a folder labeled “Frank”. Filters work by processing mail, after the system delivers the mail to you, but before you actually read it.

What do I mean by that?

How an e-mail Transaction Works

Let's start at the point that mail for you is delivered. When mail is delivered to your computer on the Internet, it arrives via a mail system which is using Simple Mail Transfer Protocol (SMTP). Although there is probably a different e-mail daemon for every week of the year, **sendmail** is by far the most popular. For the rest of this article when I speak of the “mail system” I'll just call it

sendmail, though most of what I say applies with smail, MMDF, and others. You can ignore the differences for the purposes of this article.

So, to continue, “sendmail” takes the e-mail and then decides whether the e-mail needs to be sent directly on to another system or delivered locally. If sendmail decides that the mail should be delivered locally, it goes through a short list of actions. First it looks to see if there is a .forward file in the user's home directory. If there is no .forward file, sendmail writes the mail to the user's system-wide mail file.

If the user does have a .forward file, sendmail reads the file. If the file contains an e-mail address, sendmail fowards the message to that address. If the forward file contains a pipe **|** character, sendmail runs the specified program, sending it the mail message, letting the program deliver the mail. This last case is how e-mail filters work. The sendmail daemon “delivers” the mail to your filter, and the filter delivers it (or doesn't deliver it, if you prefer) to the folders, following your set of rules. If you use the elm filter program, your .forward file might look like this:

```
"| /usr/local/bin/filter"
```

sendmail would deliver all your mail to the program called filter.

What would happen if filter was not there, or otherwise broken? We can guard against failure, by providing an alternative for sendmail.

```
"| /usr/local/bin/filter || exit 75 "
```

In this example, if the filter fails, the delivery to the user's .forward file will exit with an error number 75. This forces sendmail to back off the .forward file, and try again later instead. The **|| exit 75** only protects against catastrophe, not bad choices. If you do not correctly configure your filter, it may lose mail, but would not “fail” from the perspective of sendmail. The **exit 75** would not help you to get the mail back.

The most common place the **exit 75** can help you is when your home directory runs out of space. Most filters will gracefully fail, allowing your mail to be delivered to the system mailbox instead. This is especially helpful on systems that have disk quotas on the home directories, because the mail spool generally does not have user quotas.

Choosing a Mail Filter

There are at least four popular mail filters available: Procmail, Elm-filter, Mailagent, and MH's slocal. Procmail is a robust general purpose mail filter. By design, it is small, easy to install, and dependable. Elm is a user mail program

for reading and sending e-mail. The Elm-filter is a separate program that comes with the Elm package, and can be used with or without the rest of Elm. Slocal is the mail agent that comes with the Rand corporation's Mail Handler (MH). You might be buying the Cadillac for the cigarette lighter if you install MH just to use slocal, though. Unfortunately, Slocal does not support regular expressions (see msort sidebar for a possible solution). In contrast to Slocal, the mail filtering package called “Mailagent” supports a very rich regular expression syntax. Unlike the other filters which are written in C, mailagent is written primarily in Perl, and uses Perl's powerful regular expressions.

Procmail can write mail to “mbox” style mail files, as well as MH style mail directories. Slocal can write to “mbox” style mail files, as well as its native MH style “folders” (directories). In general, mail agents can be used interchangably with many different e-mail readers.

I use procmail to filter my mail, MH for my mail package, and “Exmh” as an X-based frontend to MH. (Exmh is written in Tcl/Tk, and is possibly the **best** way to do e-mail. [I agree!---ED])

If you are lucky, one or more of the e-mail filters described here may already be installed on your system.

Filter Installation

If you are not so lucky, you may need to install an e-mail filter. These filter programs have a varying degree of ease for installation. They all can be installed without root privileges, but save yourself some trouble, and ask your system administrator to install it for general use. If you are handy with a compiler, offer to help out. If you have to do it yourself, you'll need to get the sources, compile them, and create any needed configuration files. See the sidebar “Where to get a mail filter” for more information on getting and installing the common e-mail filters.

Spend some time figuring out your needs for mail file locking. Locking is used to guarantee that sendmail does not try to write to a file at the same time you do. The documentation for each of these packages mentions this subject, so pay attention. I found out the hard way that my mail reader was using one type of lock while my filter was using another (a moment of silence for all my lost mail, please).

After you either install a mail filter (or locate it, if it's already installed) you will need to start pushing your mail through it. As in the example above, most e-mail filters can be invoked using the .forward file. If your system does not support this feature, you can still use e-mail filters, but you may need to periodically invoke the filter using a script. You can run your script every time

you login, or once each time you read mail, or use cron to run it on a regular basis.

How a Filter Works

Now that we have a way to get e-mail pumped into our filter, let's talk about what happens next. What does the filter **do** with your mail? To start with, “E-mail filters” might be more appropriately labeled “e-mail sorters”. We all have “filters” in our kitchen: flour sifters are used to filter based on size, strainers let liquids through but not the pasta, and coffee filters let the good stuff through leaving behind the brown sludge.

Most people are not satisfied with filtering e-mail based on size, color, or fluid state. They would prefer to sort their mail based on: *Where it's from*, *Who it's from*, and *What it's about*. In e-mail terms, this might be done by looking at the **From:** or **Subject:** lines. Instead of “filter” we might do better to talk about an “agent”, “secretary”, “processor”, or even “e-mail dog”. We can give instructions to “an agent”, but not a coffee pot.

With that in mind, we can create a set of rules for our “filter” to follow. If an e-mail message matches on a rule, an action is taken. Procmail and Slocal can have only one action per rule, although there are a few ways to get around that limit. Unless otherwise directed, filtering stops after a match, and the action completes, making the order of rules important. Keep the most specific rules at the top (first), and the default case (what happens if no rules match) at the bottom. Let's look at a few examples:

Using the procmail program, we can create a file in our home directory called .procmailrc. Procmail calls its filtering directions “recipes”. Simple recipes look like this:

```
:0
* ^From.*jay@fork.com
forkmail
:0
* ^From.*cory
* ^Subject.*Elvis
/dev/null
```

The first recipe tells procmail to look for mail that contains a line starting with the the word **From**, and contains the string **jay@fork.com**. If procmail finds a match for this description, it stores the e-mail message in the file forkmail. The second recipe matches on two criteria; Email from **cory**, with a subject that includes the word **Elvis**, is deleted. The matches are based on regular expression syntax of egrep.

The equivalent using Elm's filter ($HOME/.elm/filter-rules) would look like:

```
if (from contains "jay@fork.com")  ?
        save "~/mail/forkmail"
if (from contains "cory" and subject = "Elvis") then
        delete
```

Elm's filter calls these stanzas “rules”. The matches are “egrep like”, but not as fully-featured as the matching you can get using procmail.

In slocal's .maildelivery file our rules would look like:

```
# header pattern action result string
# lines starting with a " are ignored,
# as are blank lines
>From jay@fork.com ^ "/pkgs/mh/lib/rcvstore +inbox"
>From cory         file R /dev/null
Subject Elvis     destroy N -
```

General Strategies for Filters

When constructing filter rules, err on the side of caution. Try constructing a few simple rules first. Test the rules by sending mail to yourself, and further test by leaving the filter in place for a few days. If it all works, you can start to add more complex rules. Make sure you have the “default” behavior of your filter set. Learn how to turn on debugging, and then to turn it off. Try out the logging functions. Using procmail's “mailstat”, you can get a summary of what actions procmail has taken. By investing a bit of time in figuring out how to make your filter work, you'll reap manifold time savings later.

While you are learning how to use your filter, you may want to keep a backup mail file. Instead of using one line in your .forward file which invokes your filter, you might want two lines:

```
\username
"| /path/to/filter"
```

where ***username*** is your login. This way, your mail will all be delivered to your standard system mailbox as well as through your mail filter. If you have misconfigured your mail filter, you will have a backup to retrieve your mail from. There are several other ways to do this, as well.

I find that people tend towards two schemes when building mail filters. The first technique is to sort out mail based primarily on who it is from. The second is to filter based on content or function. Mail from my manager could go in a folder called **boss**, but I get a lot of mail from him that's not particularly important, not because it's from *him*, but because of what it's *about*. For instance, mail from my manager with a subject of “Hardware budget for next year”, would better go into my “budgets” folder than my “boss” folder.

If you are not particularly worried about disk space, why not try both? Filter all the e-mail based on who it's from **and** on other factors like key words in the

Subject line or other headers. In my case, my filter could save a copy of all mail I get from my boss in the folder called “boss”, and store all e-mail with a subject line that contains the word “budget” in the folder called “budgets”.

A side effect of using filters is the excellent tracking you'll get of your e-mail. By using the logging and reporting that comes with Mailagent, Elm-filter, and Procmail, you will learn a lot about your e-mail. How much mail do you get from the “Elvis-lifestyles” mailing list? What percent of the mail is it? How does it compare to the amount of mail you get from your brother? I use the mailstat program to gauge how much time I spend supporting different departments' computers.

More Advanced Rules

PGP key handling has a special need. People who use Phil Zimmerman's PGP program to send encrypted e-mail need to exchange “public keys” to communicate securely. A common convention is to put the public portion of the PGP key in the .plan file. Anyone who wants the key can then just finger you. This is great, unless your system does not support finger, or your account is behind a firewall. The next best thing might be some sort of automatic mail response. Using procmail, you can filter for a special phrase in the subject line, like this:

```
:0 h c
* !^FROM_DAEMON
* ^Subject.*SEND-PGP-KEY
| (formail -r -A"Precedence: junk";\
   cat ~/.plan ) | $SENDMAIL -t
```

This procmail recipe first checks to see if the mail comes from the mail daemon, to prevent e-mail loops, or “ringing”. Then if the subject includes the special phrase **SEND-PGP-KEY**, procmail invokes formail which automatically constructs a reply to the sender. The e-mail that is sent back includes the contents of the .plan file. If you keep your PGP public key in this file, anybody can request a copy of your key, even if they can't finger you. The equivalent using Elm-filter would look like:

```
if ( subject contains "SEND-PGP-KEY" ) then
  execute "cat ~/.plan | mail -s \"RE: %s\" %r"
```

Elm's filter uses a macro **%s** to represent the original subject of the message, and **%r** to represent the return address.

Mail agents are really great if you have special e-mail needs. I carry an alphanumeric pager. I use procmail to watch for mail that has the magic word **PAGEJAY** in the subject line. If procmail sees the magic word two things happen. First, a copy of the e-mail is forwarded to my e-mail->pager gateway at page-

jay@pager.fork.com. When the forwarded message gets through the gateway, it will be broadcast to my pager. (The folks in my office call this “Belt-Mail”, or if the pager is set to vibrate, it's an e-mail massage.) Second, a copy of the e-mail is stored in a local folder called “pages”. Here is what the procmail recipe looks like:

```
:0 c
* ^Subject.*PAGEJAY
! page-jay@pager.fork.com
:0
* ^Subject.*PAGEJAY
pages/.
```

The first recipe uses the **c** flag. This tells procmail that even if there is a match, the matching should **c**ontinue past. In the next recipe, the belt-mail will get filed, and the matching of this e-mail will end. The equivalent in Elm's-filter rules would be:

```
if ( subject contains "PAGEJAY") then
      forward "page-jay@pager.fork.com"
      save "~Mail/pages"
endif
```

E-mail filters/agents are powerful tools that enable people who get lots of e-mail to communicate effectively. From the mundane but important task of sorting e-mail, to the complex task of responding automatically to e-mail requests, filters can deliver. Everyone involved benefits from the use of filters. You like it because your mail is always sorted the same way (*Where* did I put that CERT advisory?). The people who send e-mail to you like it because you start to respond faster by e-mail than via Canadian “First Class”. The only people who don't like it are the direct-marketing-spamers who keep trying to send you ads for “Investment opportunities”. It's OK if they get mad, because you don't ever see their mail since you installed the filter.

Any time spent learning how to configure and use the filters is won back many-fold in time-savings later.

For more information about e-mail filters, see sidebar.

Jay Allen (jay@fork.com) got his BS in chemistry from Portland State University in 1991. He is currently a lead Unix Systems Engineer at Blue Cross/Blue Shield of Oregon. Among other things, he is interested in the use of cryptography for commerce on the Internet, and secure private networks.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

XForms: Review and Tutorial

Karel Kubat

Issue #22, February 1996

Exploring XForms, a graphical user interface toolkit for X.

In this article I would like to introduce you to XForms, a graphical user interface (GUI) toolkit for C and C++ I discovered a few months ago. I've been writing programs for several platforms (including, of course, Linux) for some time now and lately have been snooping around for a good GUI toolkit for X: one that works, can be obtained free of charge, and (most importantly) one that you can use right away, without any knowledge of X intrinsics. Then I heard about XForms, downloaded it, and tried it out. I wrote my first X program in less than a day. Impressive!

If you have ever looked into one of the programmer's books about the bare X library, you understand the difficulties I'm talking about. Writing a “hello world” program using only X11 itself is no small task. Even the smallest program will require several hundreds of lines of C code.

This is where toolkits come in. A GUI toolkit is a library of functions that make the creation and maintenance of a user interface more programmer-friendly. One of the better-known packages is, of course, Motif. If you're lucky, the Motif package you buy comes with a code generator which enables you to define a simplified GUI, which is then converted into C code. If you're *very* lucky, your package has an interactive GUI builder, where you can see how your interface will look once coded. Many GUI toolkits are, however, expensive; and many GUI toolkits aren't at all friendly to a novice X-programmer.

The XForms GUI package is not only free (for non-commercial applications), but it also makes good-looking programs, has an excellent interactive design builder, comes with good documentation, and is very easy for novices to use.

Obtaining XForms

XForms is written by T.C. Zhao and Mark Overmars.

You can obtain XForms via anonymous ftp at:

ftp://bloch.phys.uwm.edu/pub/xforms andftp://ftp.cs.ruu.nl/pub/XFORMS

You should download the documentation and the XForms toolkit archive itself. The documentation is contained in a subdirectory DOC and is available in single or double sided postscript and dvi format. Since July 1995, the Linux ELF executable format is supported, as well as the a.out format. To download the toolkit in a.out format, get the file bxform-075.tgz from the directory linux. The ELF-based library is in linux/elf. The package is also available for other platforms; see the documentation.

Related WWW pages are at http://bragg.phys.uwm.edu/xforms/ and www.uwm.edu/~zhao

Finally, you can join a mailing list. Send a message to listserv@imageek.york.cuny.edu and put in the body of the mail message:

subscribe xforms *Your Name*

The authors can be reached through this list.

XForms Basics

Every program that uses XForms for its user interface manages one or more “forms”. A form is simply a window under X: with or without a border, with or without resizing capabilities, with some window title, etc. The form is a box, into which you can put “objects”: buttons, dials, input fields and lots more. Objects are what many other toolboxes call “widgets”. Each object is one of a given “class”. For instance, XForms supports the object class “button”, of which many flavors exist (a simple button, with a light-bulb on it, round ones, check buttons, radio buttons etc.). Most programs can be written using XForms' built-in classes, though XForms supports a free object class to handle special situations.

The concepts of XForms, such as classes and objects, resemble object-oriented programming (and C++ in particular). However, the XForms code is strictly C, which means that you can use it for both C and C++. What's more, XForms is built on top of the bare X11 library. Linking an XForms program requires only the presence of the XForms library, the X11 library and the math library; no other libraries are necessary.

One Input Source vs. Many

The programming paradigm you have to use with XForms (and in fact, with any X application) is somewhat different from what you might be used to. When writing an interactive program that reads its input from only one source (say, the keyboard), you are likely to use the following scheme:

```
/* forever do: */
while (1)
{
    /* fetch user's input */
    input = get_input ();
    /* determine what it is and
     * take appropriate action
     */
    if (input == one_thing)
        do_one_thing ();
    else if (input == other_thing)
        do_other_thing ();
    .
    .
    .
    /* if `quit` signaled: we're
     * all done here
     */
    else if (input == all_done)
        break;
}
```

Conversely, when you build a user interface for X, you are likely to create several input sources. For example, you can have several buttons which, when pressed, activate some part of your program's code. Input sources can even be objects that are not selectable by a user, such as a timer that runs out. The programming paradigm that represents this is implemented in X with callbacks. You create several buttons, each with its own specific callback function. The main program loop then consists of polling the events at the various input sources and activating the appropriate callback when something happens. This is the approach taken by XForms, and we will see a typical program with callbacks later on.

Code Setup

In general, an XForms program has two types of code: the user interface that uses the XForms toolkit and the code that performs some useful actions, i.e., what you want the program to do when, for instance, a button is pressed. The user interface part of a the program consists of the initializing code, the definition of the forms and commands to put the forms on-screen. Then, the program activates XForms' main loop. This loop does the event polling described above.

For example, a tiny program that only shows a “quit” button would look as follows:

```
#include <forms.h>
/* Call back function, called when `quit`
```

```
 * button is pressed
 */
void quit_button_cb (FL_OBJECT *obj, long data)
{
    exit (0);
}
void main (int argc, char **argv)
{
    FL_FORM
        *myform;        /* form to display */
    FL_OBJECT
        *quitbutton;    /* temp variable */
    /* Initializer code: */
    fl_initialize (argv [0], "XMyprog", 0, 0,
                   &argc, argv);
    /* Form definition: */
    myform = fl_bgn_form (FL_FLAT_BOX, 200, 150);
    quitbutton = fl_add_button (FL_NORMAL_BUTTON,
                 50, 50, 100, 50, "Quit");
    fl_set_object_callback (quitbutton,
        quit_button_cb, 0);
    fl_end_form ();
    /* Putting the form on-screen: */
    fl_show_form (myform, FL_PLACE_FREE,
        FL_FULLBORDER, "Button program");
    /* Main XForms loop: */
    fl_do_forms ();
    /* Never reached.. */
}
```

The definition of a function **quit_button_cb()** can be seen in this listing. This function is the callback, which is activated when the user presses the quit button. All following code is in the **main()** function: the initialization, the form definition, the placing of the form on-screen, and the activation of the main XForms loop. The initialization part is done by the function **fl_initialize()**, which connects to the X server, parses the command line, etc. In this example, **fl_initialize()** doesn't need to parse any application-specific flags; hence the two zero arguments.

The form definition is enclosed by **fl_bgn_form()** and **fl_end_form()**. Normally you do not write this code by hand, but leave it to the designer (discussed below). The form is started by stating the box type and the size of the form. Between the **fl_bgn_form()** and **fl_end_form()** statements, objects can be added to the form. An object is added using **fl_add_...()**. In this example, only one button is put in the form at the x and y coordinates 50,50 (relative to the form's lower left corner) and with a size of 100 by 50 pixels. The button text is **Quit**.

Following the button definition, the callback of the button is assigned. Normally you leave even this definition to the designer. In our case, activation of the button leads to a call to **quit_button_cb()**.

Note that to add the button, a local variable **quitbutton** is used. The same variable is used in the next statement to assign the callback. These are the only statements that require the variable; in other words, an object can be created (typically in a dedicated function generated by fdesign) without having to use a global variable. Readability and maintenance are usually improved by minimal use of global variables. Later on, you can always find the object when its

callback function is invoked—the first argument to the callback is always a pointer to the object in question.

After the form definition (terminated by **fl_end_form()**), the form is placed on the screen using **fl_show_form()**. The placement type (resizing capabilities, etc.), border type and window title are stated here. After this, XForms' main loop **fl_do_forms()** is called. This function never returns; the program terminates via the callback.

The Designer

The XForms package contains an excellent designer called **fdesign** to create and maintain a graphical user interface. The designer lets you create forms, add objects to the forms, set the objects' properties, such as colors, etc., and define the callbacks of the objects.

The designer can be finely tuned. I strongly suggest that you read the chapter in the documentation about the subject. When tuning the designer, you specify the type of the generated code, whether the designer should emit a **main()** function and/or templates for your callbacks, the stepsize in pixels of the size and placement of objects when designing a form, and lots more.

In fact, you should never modify the code the designer generates; if you want to add special **fl_...()** calls to modify the objects' appearance, do so outside of the generated code. The reason for this is that later on you may want to change the look of your program, e.g., by changing some colors. All you have to do then is to start up the designer, perform your changes, save the files and re-make. If you modify the generated code, the changes are lost once you re-save the designer file. (Of course, I came to this conclusion the hard way—I made the mistake of modifying fdesign's output.)

Installing XForms

The archive containing the XForms toolkit is currently named bxform-075.tgz, 075 being the version number. After you obtain the archive, change-dir to a sources directory (such as /usr/src) and extract the archive using **tar xvzf bxform-075.tgz**

The archive spills into a subdirectory **xforms** where you will find the Makefile. By default, a **make install** copies the library libforms.a and the header file forms.h to respectively /usr/lib and /usr/include. The designer, fdesign, goes in /usr/local/bin. If you want to use other paths, edit the file mkconfig.h before you **make install**.

If you want to see the demos, type **make**/and wait (this takes some time to complete). Then **cd DEMOS** and type **./demo**.

An Example

Having introduced XForms, I want to show you the steps to create a real XForms program. For the sake of simplicity, I present a program that manages only two forms. One of the forms has an input object, in which a user can enter a filename. This form also has a quit button to terminate the program. The other form has a browser object, in which the contents of the file are displayed. Using the scroll bar in the browser, the user can view different parts of the file.

As for the behavior of the program, we'll decide that only the form with the input object and the quit button is initially on-screen. When the user enters a filename, the form with the browser containing the file's contents appears. When an empty filename is entered, the form with the browser disappears again.

First of all, we should decide on some names. The forms will be called respectively **name_form** and **browser_form**. Inside the **name_form** there will be two objects: one input object to allow the user to type a filename and one button to quit. Both objects will need callbacks to take the appropriate actions: the functions will be respectively **input_cb()** and **exit_cb()**. These objects can remain nameless, which means that the designer will not generate global variables to address them.

The **browser_form** needs only one object in it: a browser object. This object does not need a callback since the browser will not accept user interaction (except the movement of the scroll-bar, which is handled internally by the browser). However, we will need a global variable for this browser since the callback of the input object of the first form must be able to fill the browser with the contents of a given file. We will call the browser **file_browser**.

Using the Designer

Now that we have decided on the names of forms and objects, we can design the forms. The XForms designer can be started with **fdesign design &**, telling it to save its output in the files design.fd (the design itself), design.c (generated C code) and design.h (generated header file). The ampersand starts fdesign in the background.

Once the designer comes up, click on the “New Form” button to start a new form and enter the form name **name_form**. You can rescale the window that appears to a suitable size. After this, put the required objects in the form: an input object and a button. These objects can be selected in the “Objects” menu

of the designer. Once an object is placed in the form, you can resize or move it using the mouse. The “Align” button lets you define the stepsize in pixels by which the objects are scaled or moved. Pressing F1 activates an “Attributes” window, in which the object attributes (such as font, color, callback) can be defined.

As far as the **name_form** is concerned, you should create something that looks like the screen dumps in figures 1 and Figure 2. Figure 1 shows the designer window itself, the name form, and the attribute window of the input object. (Yes, I wrote the program around noon.) In this figure the input object is “active” as can be seen by the red box around it.

Figure 2 shows the quit button and its properties.

To define the browser form, click again on “New Form” to start a second form. Enter **browser_form** as the form name, and resize the form window to your liking. Then select a “browser” object from the Objects menu and add it to the form. The form, its browser object and the properties of the browser are shown in Figure 3.

Having defined the form, we can let fdesign generate its code. Click on the “Options” button and toggle the “Alt format” entry to light up the button in front of this menu entry. This instructs fdesign to generate simple code, instead of placing form and object variables into structs. For this application, simple code suffices. Then click on the “File” button, save the file, and exit fdesign.

Putting It All Together

We can rely on fdesign to generate the code for the definition of all forms and objects. That leaves us with two tasks: the initialization code and the callback code.

The initialization code is shown in listing 1. This code defines the **main()** function in which XForms is initialized, the forms are created, the first form is put on-screen, and the main XForms loop is entered. You can see the call to **create_the_forms()**; the code of this function is generated by fdesign in the file design.c.

Listing 1: xviewfile.c

```
/* xviewfile.c -- main function of xviewfile */
#include \<>forms.h
#include "design.h"
void main (int argc, char **argv)
{
    /* initialize XForms, parse arguments */
    fl_initialize (argv [0], "XViewfile", 0, 0, &argc, argv);
    /* create the forms (fdesign-generated function) */
    create_the_forms ();
```

```
    /* show the name input form */
    fl_show_form (name_form, FL_PLACE_MOUSE, FL_FULLBORDER, "Enter a name");
    /* enter XForms loop */
    fl_do_forms ();
    /* not reached... */
}
```

Now for the second task, the callback code. The callback functions are shown in listing 2. The names of the callback functions were already mentioned in the design specifications; it is therefore important that the exact names are used when writing the functions.

Each callback activated by an object is passed two arguments. The first argument is a pointer to a **FL_OBJECT**. This argument points by definition to the object which invoked the callback. We will see how this argument is used in the **input_cb()** function. The second argument is a long int—a value which can be set to any number when defining the callback in the designer. For example, you could have two different objects invoking the same callback function but providing different numeric arguments; inside the function you could distinguish by inspecting the second argument. We won't use this approach in this application.

The callback activated by the quit button is called **exit_cb()**. This is the easy one—all it does is **exit(0)**. The callback for the input object, **input_cb()**, needs to perform more tasks. First, the name which was typed by the user must be retrieved. This is done by XForms' function **fl_get_input()**. Then, we must decide what to do with the input. As specified by the program description above, an empty name should lead to the removal of the browser window from the screen. A non-empty name should be interpreted as a request to view a file.

To accomplish this, **input_cb()** uses a static **int** variable to flag whether the browser form is yet on-screen. When a non-empty name is entered and when the browser form is not yet on the screen, **fl_show_form()** is called to show the browser. Similarly, when an empty name is entered and when the browser form is on-screen, **fl_hide_form()** is called to remove the browser form. (Instead of using an extra static **int**, you could also inspect the field int **visible**, which is part of the **FL_FORM** struct. I leave such optimizations to the reader.)

The browser itself is manipulated with browser-specific functions **fl_clear_browser()** and **fl_load_browser()**.

```
/* callbacks.c --contains the callback routines */
#include <\<>forms.h>
#include "design.h"
void exit_cb (FL_OBJECT *obj, long data)
{
    exit (0);
}
void input_cb (FL_OBJECT *obj, long data)
{
    char const
        *name;                              /* entered filename */
```

```
    static int
        browser_on_screen = 0;              /* is browser on-screen yet ? */
    /* determine the entered name */
    name = fl_get_input (obj);
    if (name && *name)                      /* a name was entered */
    {
        if (! browser_on_screen)            /* make sure browser is there */
        {
            fl_show_form (browser_form, FL_PLACE_CENTER,
                          FL_FULLBORDER, name);
            browser_on_screen = 1;
        }
        fl_clear_browser (file_browser);    /* clear previous contents */
        fl_load_browser (file_browser,      /* load in file */
                         name);
    }
    else                                    /* empty input was given */
    {
        if (browser_on_screen)              /* remove browser from screen */
        {
            fl_hide_form (browser_form);
            browser_on_screen = 0;
        }
    }
}
```

Finally, no program is complete without an automatic maintenance description in a Makefile. Here is an example:

```
# Makefile -- makefile for xviewfile.
# Used objects:
OBJ = xviewfile.o callbacks.o design.o
# Compilation flags:
CFLAGS = -c -O2 -Wall
# How to make the program:
xviewfile: $(OBJ)
        $(CC) -o xviewfile $(OBJ) -lforms -lX11 -lm
-s
# How to clean up the mess.
clean:
        rm -f $(OBJ)
```

Loose Ends

The listings show only the tip of the iceberg of as far as XForms' possibilities are concerned. For the sake of brevity, I did not address subjects such as color definition or font selection.

One thing you will want to try is making resizable forms. Whether a form can be resized is defined in the call to **fl_show_form()**, which displays the form. How the form is resized is defined in the form definition itself. Again, you should use fdesign to set these options. For example, you might have a form with a button and an input object at the top, and with a browser below thrm. When such a form would be resized, you'd probably want only the browser to expand or shrink, not the button and input object. Such capabilities are set with objects' resizing options and gravities. XForms, and hence the designer, supports “object groups” for this purpose: you can group objects and define the resizing options and gravities to apply to a whole group.

Another useful property of any X program is the ability to let the user overrule settings of the program via command line flags, resources set via xrdb, or resources set via an application defaults file. XForms supports all of these; **fl_initialize()** can be used in combination with **fl_get_resources()** to scan for meaningful flags or resource settings. Implementing the resource recognition is not entirely trivial but goes beyond the scope of this article. If you're interested, look at some of the programs built using XForms. The XForms distribution as well as the WWW resources contain pointers to useful and illustrative programs.

Naturally, there are lots of other useful XForms possibilities. I leave it to you to read through the excellent documentation on a quiet evening with a glass of (virtual) beer.

Review of Xforms

Needless to say, I am very enthusiastic about XForms. The setup of the XForms toolkit (using forms, objects, classes) is very intuitive even when you have no previous experience with X programming at all. You do need good knowledge of C, though.

The range of object classes in XForms is so wide that I haven't (yet) needed to define a free object class. If you plan to write everyday programs, XForms probably has the classes to suit your needs, and the classes have plenty of dedicated functions to make the objects appear the way you like.

As far as the designer is concerned, I have come to cherish this tool, even though I first regarded it as just a flashy extra. The ease with which you can adapt a program to perform an extra task, by expanding an existing form to hold, say, an extra button, is incredible. No more code editing to adjust the coordinates of all widgets!

There are, of course, drawbacks. I think it's a shame that XForms is not distributed with the full sources. The fact that XForms can be used free of charge for non-commercial applications is of course a huge advantage, but still, I'd rather have the sources at hand. This irritation can be overcome if you're willing to pay the price: the authors do sell license agreements that include the source files.

Another drawback is that if you do not use the ELF-based version of XForms, you have to link your programs against a static libforms.a. There is no shared a.out-based library for XForms. Therefore, unless you migrate to the ELF executable format, even the smallest program turns out to be about 130KB. (On the other hand, I also use statically linked Motif programs. The 130KB is, by comparison, trivial.)

The XForms library is “nice” to programmers, which is an advantage when you are a new user or when you're developing a program. For example, when a program tries to remove a form from the screen which is not on-screen in the first place, XForms will pop up a warning window. You are then presented with three choices: to ignore the action, to abort the program, or to suppress such warnings. This feature is very handy when you are testing a program. However, such safety nets add “superfluous” code to a program. Once a program has been tested, such precautions are, in my opinion, no longer necessary and therefore only a burden. Ideally, I'd prefer two flavors of the XForms library: a developer's version with the full safety features and a production version without them. On the authors' part this would require only some **#ifdef/#endif** compilation directives in the code.

On the whole, I find that there are many more advantages to XForms than there are disadvantages. I use XForms now and I'm planning to continue to do so. I still haven't hit a wall in my X programming efforts that could not be overcome using XForms' own features. And I haven't even explored all possibilities of XForms yet.

Karel Kubat lives in the northern part of the Netherlands and is currently working on his PhD thesis. He is a Linux fanatic and therefore pretends to know everything about the subject, since that's what being a fanatic is all about. He can be reached via e-mail at the University of Groningen at karel@icce.rug.nl.

Archive Index Issue Table of Contents

Advanced search

Open Systems World/FedUNIX '95

Kevin Pierce

Issue #22, February 1996

The show included the second annual Linux Conference, sponsored by *Linux Journal*, which featured a day of informative sessions and tutorials and a one-day class for novices and intermediate Linux users.

Open Systems World/FedUNIX '95 was held the week of November 13 at the Washington Convention Center, Washington, DC. The show included the second annual Linux Conference, sponsored by *Linux Journal*, which featured a day of informative sessions and tutorials and a one-day class for novices and intermediate Linux users. The first day covered several topics, including an introduction to Linux, building a World Wide Web site with Linux, and porting to Linux. The one-day class, entitled “Linux for the New User”, was taught by the Editor of *Linux Journal*, Michael K. Johnson.

The trade show ran for two days following the Linux conference. Most of the Linux-related companies were in the same area and received a lot of traffic. The overall feel of the show was that Linux was one of the “hottest” topics and that we needed to work together to ensure the success of the “better” system.

Caldera had one of the largest Linux-related booths and was situated towards the center of the show floor. Open Systems World also set up a 40'x40' “Linux-Lounge” next to the Digital booth consisting of three cocktail tables surrounded by chairs where people could sit and talk about Linux. Whenever I passed the Linux Lounge, it was packed with people doing just that.

Although Open Systems World/FedUNIX occurred in Washington, DC during a federal government shutdown and the same week as Comdex in Las Vegas, it looked as though the Linux Conference was a success, and we are looking forward to next year.

Advanced search

COMDEX '95

LJ Staff

Issue #22, February 1996

Open Systems World in DC had a Linux track and exhibition at the same time as Comdex and drew many of the Linux Vendors who otherwise might have been at Comdex.

COMDEX is the second largest computer trade show in the world, offering multiple convention floors with 2,000 exhibitors plying their new computer products to approximately 200,000 attendees in Las Vegas, Nevada in November of 1995.

Open Systems World in DC had a Linux track and exhibition at the same time as Comdex and drew many of the Linux Vendors who otherwise might have been at Comdex. A few Linux Vendors were, however, represented at Comdex: WorkGroup Solutions, DIOS, InfoMagic, Walnut Creek CD-ROMs, and Linux International.

Linux International is a not-for-profit organization formed to promote Linux to computer users and organizations. Mark Bolzern of WorkGroup Solutions and Belinda Frazier of SSC helped coordinate Linux volunteers at the Linux International Booth. Belinda and Mark offer their thanks for the energy, enthusiasm, and hard work of the Linux International Booth volunteers: Tom Lang, Bob O'Connell, David Mills, Charles (Russ) Lyttle, Joanne Wagner, Jon Gross, Vassili Leonov, Kevin Chau, Ralf Schmidt, Bjoern Roth, Roland, Baer, Joel Sager, Karlos Smith, Michael Hawes, Frank Keeney, Allison Keeney, Scott Rasmussen, David Harlan, Rob Flickenger, Bryan Thompson, and Mark Samarraie.

At the LI booth, the response to Linux was overwhelmingly positive. Questions ranged from “I've heard a lot about Linux, but I'm not sure what it is. Can you enlighten me?” to “I haven't checked for a few days—what is the latest development kernel?”

There was a certain “underground” feel to the Linux booth, surrounded as we were by the large, flashy, expensive displays of the Windows and Mac worlds. This was reinforced by the people walking by, giving us the thumbs up and saying, “Glad to see you guys here—I love Linux!”

Comdex attendees tend to be small or medium business owners, and we fielded a large number of questions regarding the suitability of Linux in a small to medium sized business setting, often spending a lot of time discussing the integration of Windows machines with some sort of central server (Linux of course!).

We also talked to a number of Internet Service Providers (ISP) who were interested in moving from their current setup (often Windows NT) to something that was cheaper.

BYTE Magazine named WorkGroup Solutions WGS Linux Pro 3.0 Runner Up for the the “Best of Comdex Award” in the category of operating systems. The winner was the Apple Newton 2.0. We hope Linux will emerge the winner next year.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #22, February 1996

Readers sound off.

Why So Slow?

Questions:

1. How come your January issue is on the Web and I still haven't received my December issue? Did it get lost in the mail or am I impatient?

2. Can you put more articles on-line for us paying subscribers? I paid my dues and feel like I am due the flexibility of reading online *or* on paper, please.

—RSVPAdam Holt holt@mit.edu

"Circumstances beyond our control"

1. The December issue was mailed one week late due to a problem at our printers, which set back printing, and therefore mailing, by one week.

The Table of Contents is posted to the WWW and Usenet as soon as an issue is sent to the printer; printing and mailing take another three to four weeks.

2. We are working on putting articles on-line, but our first priority is putting the magazine on paper. If we don't get the magazine on paper, there won't be one on the WWW, either. We are not over-staffed, and right now many of the articles you see already on our site were html-ized by volunteers. Our goal is to have the entire journal on on the WWW, and we are working towards that goal by creating tools that will allow us to work simultaneously on the paper and WWW versions of the articles. But that's another project which competes for time with the all-important process of getting the articles on paper.

We understand that you feel that we owe you having all the articles on-line; but it's not something we offer as part of the subscription, partly because we don't have the resources to provide that service in a timely fashion. We would like to suggest that you compare the WWW service that we provide now, with a full, interactive index, many of our previous articles, and response forms all available, with what we provided a year (and less) ago—no WWW service at all. Please understand that we are growing, and increasing our level of service as we grow, but it takes time.

New Hungry Programmers Location

I'm just writing to give you the current URLs and e-mail addresses for the Hungry Programmers stuff that were mentioned in the Lesstif and Viewkit article in the November issue of *LJ*.

The URL for the Hungry Programmer's homepage is now http://www.hungry.com:8000/

The URL for the lesstif stuff is now http://www.hungry.com:8000/products/lesstif/

The URL for the lesstif documentation project is now www.hungry.com:8000/products/lesstif/Lessdox/LessTif.html

The URL for the viewkit stuff is now http://www.hungry.com:8000/products/viewkit/

The e-mail address for the hungry programmers is now hungry@hungry.com

The address for the mailing list is also different than what was published in the article. It is now majordomo@hungry.com.

—Thanks, Chris Toshok toshok@hungry.com www.cs.uidaho.edu/~toshok

Error found . . .

An alert *LJ* reader, Robert Day, pointed out an error in the Find tutorial published in the December issue. The tutorial provided an incorrect example for using find to locate files with the SUID bit set.

The example was:

```
find / -perm 4000 -print
```

As shown earlier in the article, this command would find only files whose permissions are **exactly** 4000. The correct example is:

```
find / -perm -4000 -print
```

That's it.

Archive Index Issue Table of Contents

Advanced search

DECUS conference in San Francisco

Phil Hughes

Issue #22, February 1996

While our booth was the only Linux-specific one, there was a lot of Linux interest.

Our first coverage of a DECUS conference was in the August 1994 issue where we reported on the May 1994 DECUS conference in New Orleans, at which Linus Torvalds spoke. In May, 1995, *Linux Journal* was represented at SSC's booth at the Washington DC DECUS '95 conference. That booth was quite a success so we decided to see how Linux would do at San Francisco.

The trade show ran three days, Monday to Wednesday, December 4th to the 6th, with the whole conference starting on Dec 2 and continuing through the 7th. DECUS scheduled two four-hour Linux talks: “How to write a Linux Device Driver” by Michael K. Johnson and “Linux: The Open System for Everyone” presented by me. In addition, while not part of the formal DECUS show, a special Linux Daze was scheduled to start on Wednesday afternoon and continue through Thursday.

The Trade Show

While our booth was the only Linux-specific one, there was a lot of Linux interest. As at the trade show in DC, we were the most popular booth in the “low-rent” district of booths. There were quite a few people who hadn't heard of Linux, but there were even more who had and were happy to see *Linux Journal* at the show. We gave out over 1000 copies of our December issue. Not bad for a show with attendance of less than 2500. We also gave out bumper stickers and sold Linux and other SSC products. Sales of T-shirts and Linux CDs were higher than expected.

We were also showing Zebu Systems' Orion Firewall Systems. Quite a few companies at the show were shopping for firewall systems. The SSC booth was staffed by an assortment of people with varying areas of expertise. Carlie

Fairchild provided information on SSC's Linux products and was in charge of the overall booth operation. Michael K. Johnson and I were both in the booth to address *Linux Journal* questions as well as general Linux questions, as was Randolph Bentson, the author of the Cyclades driver software, and author of a soon-to-be-published Linux book.

David Bonn and Larry Stelmat of Mazama Software Labs were also at the booth. David is the president of Mazama which is a Linux-based software development company in the San Francisco Bay area and the Puget Sound (Seattle) area; David is the primary author of the firewall software that runs on the Orion Firewall System.

A walk around the trade show and talk with other vendors showed, once again, more interest in Linux. AT SSC we use the Progress database. It is currently running on a Unix 5.4.2 xystem because there is no Linux port available. I did a little Linux evangelism at the Progress booth and they admitted that there were other customers asking for a Linux port. If you are a Progress user, I encourage you to voice your interest in a Linux version. Seems like they're starting to hear us.

On Carlie's search for a power cord (you always have to forget something at trade shows), she met Jan Cherkowski, President of Advanced Business Technology Inc. Jan is an avid Linux enthusiast and wants to ship Linux/Alpha solutions. Don't be surprised if you see his ads in a future issue of *Linux Journal*.

Another hardware vendor, Polywell, showed similar interest in reselling Linux/Alpha systems. Because I was wearing a Linux T-shirt Emmett Kiest-Jones of Magic Software Enterprises Inc. came running after me to talk to me. Magic offers a RAD (Rapid Application Development) environment currently available on VMS and other platforms. Emmett said there was significant interest in producing a Linux board.

The people in the booth next to us were the publishers of *Digital UNIX News*, a bi-monthly on Unix for Digital platforms. In their current issue there is an article on Linux which initially appears to be very favorable, talking about many of the reasons Linux is a good system. It then ends with a conclusion that doesn't seem to follow—that Linux is a toy.

The Linux Sessions

On Wednesday, there were two Linux sessions as part of the standard DECUS Program. The first one, “Linux: the Unix for everyone”, was my session. While attendance was low (10 students), interest was high and the talk went very well

(if I do say so myself). The afternoon session was “Writing a Linux Device Driver” by Michael K. Johnson. Michael was pleased with the participation in his session as well.

Linux Daze

Linux Daze officially started with a room of Linux vendors at noon. The vendors included Caldera, Cyclades, Digital, O'Reilly, SoftCraft, NekoTech and BVC.

At 7pm there were two talks on Linux, as part of Linux Daze. I gave a short talk called “Linux in Publishing”, and Matt Welsh spoke on “Porting to Linux”.

Thursday, Linux Daze continued with a full day of vendor exhibits (much better attended than those on Wednesday afternoon) and a full day of sessions on a variety of Linux topics. The day was wrapped up with CD-ROM giveaways, and topped off with a drawing for the grand prize—a complete Alpha System (Digital's **Universal Desktop Box**) donated by NekoTech.

DECUS was a success for Linux. While attendance was low, Linux interest seemed high. Some scheduling problems and lack of communication and promotion of the Linux events seemed to have resulted in the loss of a few attendees but I expect interest in Linux to remain high in the Digital community. Now on to the Feb 26-March 1 DECUS in Vancouver, BC to see how Linux does there.

Archive Index Issue Table of Contents

Advanced search

System Administration: Maximizing Linux Security: Part 2

Æleen Frisch

Issue #22, February 1996

Last month's installment covered many aspects of password protection. This month's installment goes on to explain several other aspects of system security.

File-system Protection

Besides password protection, file ownerships and protection modes are the other major component of traditional UNIX security. Although far from a perfect solution, a carefully set up and maintained file-system will provide a great deal of protection from harm in the event that a non-root account is compromised. File-system security includes these considerations:

- Correct ownerships and protections for system files: command binaries, shared libraries, and so on. Such files generally should be owned by root or another system user like bin and by a system group such as system or bin. They should not be group, or world, writable. Some files, such as the shadow password file, should be restricted to owner access only.
- The same considerations apply to the directories where system files are stored. Keep in mind that write access to a directory allows you to modify any file within it regardless of the ownership or protections of the individual file (although SETGID access on a directory restricts users' access to their own files, as for /tmp).
- SETUID and SETGID files deserve special scrutiny. They should be kept to a minimum and be thoroughly tested prior to installation. Any newly appearing SETUID root files should be regarded with extreme suspicion. The following command will locate all SETUID and SETGID files on the system:

```
# find / -type f \
\( -perm -2000 -o -perm -4000 \) -ls
```

- User home directory trees should also contain no group, or world-writable subdirectories.

- User login and shell configurations files—.login, .profile, .cshrc, and so on—should also be owned by each user and writable only by their owner.
- Group memberships must be designed with care so that users are given access only to those files and directories they need.
- Any tampering with system binaries, libraries, configuration files, and other vital data must be detected right away.

All of these can be summarized into just two overriding principles:

- Know what normal is (and get your system to that state). This is made somewhat more difficult on Linux systems because there is considerable variation in system file ownership among the various distributions, so you will have to make some of those decisions yourself.
- Make sure it stays that way by continuously monitoring it and performing regular backups.

The Tripwire facility from the COAST project at Purdue University can take care of the second step. Tripwire can record the correct state of the file-system and then, some time later, compare the current configuration with the saved one and report on any differences. It can consider external attributes of a file such as its ownership, protection, size, inode number (this would change if a file were replaced using standard UNIX commands), inode creation date, and file modification date.

However, since it is possible to modify a file without changing any of these items, Tripwire also calculates a number of file signatures for each file. In general, a file signature is a value computed using the contents of the file. Tripwire can compute file signatures using up to 10 different algorithms of varying complexity and corresponding difficulty in forging.

While it is possible to alter a file and still retain the same file signature for a single algorithm—in fact, it is relatively easy to do so for lower quality file signatures such as traditional checksums—altering a file without changing two or more different file signatures is a very hard problem indeed. When Tripwire checks the file-system, it can compare multiple file signatures for each file, thereby virtually ensuring that any alteration will be detected.

Tripwire has been ported to Linux, and it builds easily. After you have finished compiling the executables, it is important to run the test suite the package provides to ensure that everything is operating properly. The following command initiates the test suite:

```
# make test
```

To get started, you first run Tripwire in its initialization mode (**tripwire -init**). It is essential that you do so on a system known to be clean; ideally, Linux will have been reinstalled from the original media. In this mode, Tripwire creates a database listing the current attributes and file signatures you have requested for the files specified in its configuration file. During this initial run, you should compute as many different file signatures as you have CPU cycles to apply, including at least two different highly secure algorithms. You should also set up the configuration file to include as much of the system as possible, so you'll have data even for files you won't necessarily be watching on a regular basis should you ever need it.

The database will need to be similarly updated whenever an operating system upgrade occurs (given the rebuild rate for Linux kernels, that could be pretty often on some systems). Once the database is created, it must be stored in such a way that it cannot be tampered with under any circumstances (otherwise, a hacker could, for example, replace a file and also alter the information corresponding to it in the database). The Tripwire documentation suggests placing it on physically write-protected media, such as a locked diskette or removable disk, which is taken out of the drive when it is not in use. When the database is protected in this way, even changes made from a compromised root account can be detected. If possible, the Tripwire software itself should also be similarly protected.

After the initial database is created, Tripwire may be used to check the integrity of the file-system. How regularly you run Tripwire in this mode depends on the needs of your system and site, but I would recommend doing so nightly if at all possible. Figure 1 gives an example of the sort of report that Tripwire produces.

```
deleted: -rwxr-xr-x root  77828 Aug 23 22:45:43 1995
/usr/bin/refer
added:   -rwxr-xr-x root  10056 Mar 19 12:33:11 1995 /etc/passwd.save
changed: -rwxr-xr-x root 155160 Apr 28 15:56:37 1995 /usr/bin/perl
### Attr        Observed (what it is)         Expected (what it should be)
### ========== ============================= =============================
/usr/bin/perl
st_size: 155160                         439400
st_mtime: Fri Feb 17 12:10:47 1995      Fri Apr 28 12:33:11 1995
md5 (sig1): 1Th46QB8YvkFTfiGzhhLsG        2MIGPzGWLxt6aEL.GXrbbM>
```

On this system, the executable for the refer command is missing, a new file (/etc/passwd.save) has appeared (from Tripwire's point of view, anyway), and the executable for Perl has changed size, modification time, and file signature (computed with the MD5 algorithm). All these changes are important and should be investigated, although none of them conclusively indicates unauthorized activities (refer could have been deleted accidentally, /etc/passwd.save could have been created as a backup by a system administrator, and Perl could have been rebuilt since Tripwire's database was last updated).

Tripwire's configuration file (conventionally named tw.config) is very flexible and allows you to specify exactly what files and directories are checked and what attributes and/or file signatures are compared, in as much detail as you like.

Tripwire does an excellent job of monitoring the file-system for any changes. However, there are other system functions that also bear watching. The Computer Oracle and Password System (COPS) performs several useful tests of system security, and I recommend obtaining it and running it regularly. COPS is most useful for checking the following items:

- The syntax and content of the password and group files.
- Anonymous ftp setup.
- User environments: umask values and PATH variable definitions as defined in users' login configuration files.
- Searches for known-to-be-insecure versions of commands by comparing the dates of system executables with data from CERT advisories.

In addition to Tripwire and COPS, the following other facilities can be very useful for system security monitoring:

- The /var/adm/sulog file, which contains records of each use of the su command (successful and unsuccessful). It should be examined regularly.
- The syslog facility: many subsystems log messages via syslog. Its configuration file, /etc/syslog.conf, specifies what types of messages are recorded as well as their destination log file.
- Data gathered by the optional accounting facility can be useful for some kinds of detective work. In order to use this subsystem, you will need to install the accounting and quota kernel patches, rebuild the kernel, and compile programs in the accounting utilities package.

The Computer Incident Advisory Capability (CIAC) has created the Merlin program as an easy-to-use graphical front end to several security monitoring packages including COPS, Crack and Tripwire.

Improving Network Security

So far, we have focused on security issues that arise in the context of a single computer system. However, most systems are connected to networks, and so must deal with the variety of threats which arise from that context.

Standard TCP/IP offers only limited mechanisms for controlling network access, and they are designed for convenience of access rather than true network security. In its standard form, the /etc/hosts.equiv file contains a list of systems

trusted by the local host, and users with the same user name on one of these remote systems can log into the local system via the network without having to provide a password.

Under normal circumstances, this makes sense. The problem comes when an account on a remote system has been compromised; trusting that system then puts your own system at risk as well. Of course, this problem is magnified many times when it is a remote root account that has been broken into. For this reason, it is a very bad idea to allow password-less root access between systems, and the /etc/hosts.equiv mechanism does not do so (however, the other method for setting up account equivalence does),

Each entry in a .rhosts file contains a hostname and (optionally) one or more user names. It allows password-less access to the local account from the listed remote user names (or to a user with the same user name, if the entry contains only a hostname). For example, when the following .rhosts file is in user chavez's home directory, it will allow user vasquez on hamlet and user chavez on romeo to log in to her account via the network without a password:

```
hamlet vasquez
romeo
```

The problem with the TCP/IP notion of trusted systems is that trusting another system has implications beyond the interactions between the local system and the trusted remote system. Inevitably, trust operates in a transitive fashion: if system A trusts system B, and system B trusts system C, then to some extent, system A trusts system C whether it wants to or not and whether it knows it or not. Such chains can go on indefinitely, and hackers are notoriously good at exploiting them.

Accordingly, the following precautions will minimize the risks of trusted network access to your system:

- Include the minimum number of hosts in /etc/hosts.equiv.
- Make sure there is no [cw]+[ecw] entry in hosts.equiv (this acts as a wildcard and trusts every system in the universe).
- Use the **-** entries supported by the Linux version of this file as appropriate. A hostname preceded by a minus sign in /etc/hosts.equiv requires a password from every user from that system who wants to log in to the local system. It also nullifies any entries for that host in all .rhosts files on the system. Thus, this mechanism serves as a ways to override transitive trust.
- The Linux version of /etc/hosts.equiv allows hostnames to be optionally followed by a username, which gives that user password-less access to

any non-root account on the local system. I don't recommend using this feature.

- No /.rhosts file should exist on the system; remote root access should always require a password (if it is allowed at all).
- Monitor users' ~/.rhosts regularly for inappropriate entries.

Finer Control Over Network Access

The traditional hosts.equiv mechanism allows for only the crudest level of access control. The TCP Wrappers package, which is included with nearly every Linux distribution, provides for much more detailed control over which remote hosts use what local network services, as well as the ability to track and record network-based system access.

The TCP Wrappers package provides the tcpd daemon which introduces an additional layer between inetd, the primary TCP/IP daemon, and the score of subdaemons that it manages. Daemons for services like the telnet facility are started on an as-needed basis by inetd; once TCP Wrappers is installed, requests for telnet services go to it first and are granted only if the system configuration allows it.

Once the tcpd daemon is built, you simply modify inetd's configuration file, /etc/inetd.conf, placing any and all of its subdaemons under tcpd's control. For example, Figure 2 shows how to change the entry for the telnet daemon to use TCP Wrappers.

```
#service socket protocol wait? user  program         arguments
telnet   stream tcp     nowait root  /usr/sbin/in.telnetd
```

is changed to:

```
#service socket protocol wait? user  program         arguments
telnet   stream tcp     nowait root  /usr/sbin/tcpd  in.telnetd
```

Access to network resources is controlled by TCP Wrappers' configuration files, /etc/hosts.allow and /etc/hosts.deny. The first file contains entries specifying which hosts may use which services:

```
in.telnetd : hamlet romeo
in.fingerd : LOCAL EXCEPT juliet
```

The first entry says that telnet requests from hamlet and romeo will be honored, and the second entry says that remote finger commands may be run from any local system except juliet (a local system is defined as one not containing a period in its name).

The hosts.deny file contains entries denying specific services:

```
netstat  : fool
in.tftpd : ALL
ALL      : ALL
```

The first entry denies the use of the netstat service to host fool, the second entry disables the trivial ftp facility, and the third entry acts as a catch-all, denying everything that hasn't been explicitly allowed to everyone.

When tcpd considers a request for network services, it uses the following process:

- If hosts.allow authorizes its use, the request is granted. The first applicable line in the file is used.
- If no entry in hosts.allow applies, then hosts.deny is consulted. If that file denies the service, the request is denied, and again the first applicable line in the file is used.
- If no entry in either file applies, the request is granted (note that an ALL:ALL entry in hosts.deny prevents this case from coming into play).

TCP Wrappers logs its activity to the syslog subsystem via its daemon facility. It generates lots of data which can be cumbersome to examine manually. The swatch package provides a useful way of automatically sorting through any output stream for events you specify in advance, and it is very useful in conjunction with TCP Wrappers.

Probing Network Vulnerabilities

The network should be examined for potential security problems on a regular basis just like the local system. The once notorious Satan program provides one way of doing so. Satan is designed to look for network vulnerabilities from the outside in. It looks for a variety of problems, including:

- The availability of notoriously insecure network services, such as rexd and old versions of sendmail.
- The setup of any ftp and/or tftp facilities.
- A variety of NFS vulnerabilities.
- X server vulnerabilities.
- Unprotected modems.

For those concerned about the misuse of such a powerful tool, the Gabriel and Courtney packages attempt to detect suspicious uses of Satan itself.

Don't Despair

We've covered a wide variety of threats to system security in this article. Don't let the sheer number of them overwhelm you. All you can do is protect your

system as well as is currently possible and make frequent backups so that you can recover quickly in the event that your best efforts are not enough. Remember that system security is an ongoing process, not something you can take care of once and then forget about. And as in all of life, there are no guarantees.

See sidebars for security resources or more information about system security.

Æleen Frisch (aefrisch@lorentzian.com) manages a very heterogeneous network of Linux and other UNIX systems and PCs. Having recently finished second editions of two books, she looks forward to pursuing her true calling: pulling the string for her cats, Daphne and Sarah.

Archive Index Issue Table of Contents

Advanced search

Caldera Network Desktop Preview II

Eric Goebelbecker

Issue #22, February 1996

It provides an interface and value-added features that could help Linux reach a wider market.

The Caldera Network Desktop is an operating environment based on the Linux operating system and bundled with commercial software that is licensed and/or developed by Caldera, Inc. It provides an interface and value-added features that could help Linux reach a wider market. In addition to the important features offered by the Network Desktop, Caldera has announced they will be offering additional products for Linux, including a version of WordPerfect 6.0 native to Linux and priced competitively with Windows applications. For these reasons, the Network Desktop is an important product for the Linux community, and it deserves some attention.

The Network Desktop offers an installation and software management system that makes it easier to manage and install for end users and system administrators. The package management system, which is Red Hat Software's **RPM**, is superior to any software management system I've ever seen on any commercial Unix or DOS/Windows environment. It also provides a network backup utility with a graphical interface, automation capabilities, a distributed database and compatibility with other versions of UNIX running the same tool. Caldera also bundles in a font server that further simplifies managing enterprise networks and supports font formats normally not available with Linux.

For greater interoperability, Caldera has licensed Netware 4.x client code directly from Novell. They have developed a netware automounter and a full access to Netware 4.x directory services.

Caldera also includes precompiled and configured internet tools such as a web browser, web server, configurable FTP server and an important TCP security tool.

And, of course, Caldera also provides the desktop shell featured prominently in their advertisements.

I installed the Network Desktop Preview II on a 90MHz Pentium system with 24MB of RAM and two 525MB IDE hard drives. The Preview II release is based on Red Hat Commercial Linux Release 2.x, which includes Linux kernel version 1.2.13, XFree86 version 3.1.2, and gcc version 2.7.0.

Caldera's (and to be fair, Red Hat's) installation procedure is definitely one of this product's strong points. Installing the Network Desktop is arguably easier than installing Windows 95 or OS/2. This is because of two important features: Caldera's documentation and Red Hat's installation program and package management tools.

One way Linux can reach a wider market is by becoming more accessible to the “average” user. While many of us long-time Linux users consider the discovery process involved with getting a new CDROM or downloading new software from the Net part of the fun, many users who would love to learn more about UNIX and computing don't have the time, skill, or patience that is sometimes necessary to build and maintain their first Linux system.

The Network Desktop installation procedure addresses this issue beautifully. Since I've never had a Red Hat Linux CD (a circumstance that I do plan on changing), I don't know where Red Hat's installation ends and Caldera's modifications begin. Regardless, the final product is a pleasure.

Caldera's **Getting Started** guide has eight pages on how to install the Network Desktop on a PC. After covering the hardware requirements, the chapter gets at the information necessary to continue the installation by asking the installer a series of questions about his target PC. Installers are instructed to record their answers; these responses constitute the information regarding network configuration necessary to successfully complete the process.

Following this section is an excellent discussion about memory requirements, booting multiple operating systems, and several other topics new users need to be familiar with. After covering this information, the installer must choose the appropriate kernel.

The manual contains a few charts and a table that find the right image based on the network card, SCSI adaptor and CDROM; these are especially useful to someone installing Linux for the first time. The installer is led through the process of creating 4 floppies for building the system. This is a process common to just about every recent Linux distribution.

For people upgrading from an earlier release of Linux, Caldera offers something very convenient. The CDROM contains a set of PERL scripts that prompt the installer for the information necessary to select a kernel. The scripts save the XF86Config file and mouse device link (for X-Windows), the fstab (for file systems) and the workstation IP address. If there is no Ethernet card, the script warns the upgrader that PPP/SLIP configuration information will not be saved. The scripts then create the floppies.

The process of booting from the floppies and preparing for the transfer of the software from CD is pretty routine. The partitioning process is smooth and well-documented, and as I said earlier, Caldera does a fantastic job of explaining the process of booting multiple operating systems.

I did find one possible problem for some users later in the install. I do not have an Ethernet card in my system, and since I only use PPP for networking, I doubt I am in the minority.

The network configuration portion of the installation prompts the user for the necessary information, saves the information to the system configuration files, and attempts to configure the first Ethernet interface. If there is no interface, the procedure displays an error. To the user, the failure indication looks like the entire network configuration process has been aborted, when in fact the only error is that the interface could not be set up.

There is no option to skip configuring the Ethernet interface while still entering the other network information such as the host and domain name. In addition, the boot scripts attempt to configure the nonexistent interface during every restart. Since the installer is given the choice of selecting a kernel image with no Ethernet drivers, this doesn't make sense. There is also no way to skip configuring networking completely and still come out of the process with a hostname.

Since the Network Desktop ships with so many Internet tools, and since Caldera also provides the bulk of their support and documentation over the Net (more on that later), I expect that many users will want to use it to dial into the Internet via PPP and SLIP. Automated configuration tools for **dip** and **pppd** would be a welcome addition.

After entering the configuration parameters and selecting the disk partitions, the user can either choose the express installation or opt to install each package separately. To select the packages individually, the installer uses **glint**, the Red Hat package tool, which I will cover later.

When the software installation is done, the install program will configure X-Windows (if the user doing a first-time install rather than an upgrade) and also configure LILO. I was quite impressed with the X-Windows automatic configuration.

In addition to the thorough installation instructions and explanation of basic concepts. Caldera's *Getting Started* guide provides a wealth of information for new as well as experienced users. The introductory chapters also cover the GNU General Public License (GPL), which components of the Desktop are redistributable, what conditions they can be redistributed under, and which components are copyrighted. The full text of the GPL and other licenses are included in one of the appendices.

The *Getting Started* guide also explains how to get support for the Desktop. Another of the unique features of Caldera's Network Desktop is that Caldera plans on taking full advantage of the Internet as a primary mechanism for supporting clients. The ntroduction lists all of the necessary e-mail addresses and information about Caldera's Web site.

In the appendices, Caldera covers the Linux File System Standard (FSSTND), explains how their product deviates from it, and gives a representation of the kernel source tree. They also cover what users and groups are created, and how Red Hat Linux uses the System V init scripts to start and shutdown subsystems.

In addition to the *Getting Started* guide, Caldera supplies a large amount of documentation in HTML. This information is tied together with a Caldera_Info page that appears as an icon on the desktop. There are also links to Caldera Web Server. I really can't stress the value of Caldera's and Red Hat's documentation enough. The information available to a user of the Network Desktop is broad and abundant.

The Desktop Application itself is a File Manager/Finder like shell for X-Windows. However, it doesn't just act as a “launching pad” for existing programs; Caldera provides utilities for adding and removing users, setting the system time, and managing file-systems as well. A screen capture of the User Configuration tool screens is shown in Figure 1. The tools make tasks such as adding printers and filesystems much easier. The most important, however, is the package maintenance tool, **glint**.

Figure 1a: Group Manager Screen

Figure 1b: Edit User Screen

Glint facilitates the addition or removal of software from an existing system. Figure 2 shows the initial display when glint is activated. Red Hat packages are organized in hierarchies. For example, X11 is a “root” hierarchy. Under it are Applications, XFree86, Libraries, etc. Under XFree86 is Servers and then finally some packages, such as XFree86-fonts, XFree86-devel and XFree86 itself. Each of these packages also has a version number that is used to identify upgrades.

Figure 2. Initial glint Screen

Figure 3 displays what an installer sees when the font package is selected and the Query button is clicked: a description of the package is displayed with a list of all of the files contained in it. From here the package's contents can be verified if a problem is suspected or if a user needs to know what files are installed on a workstation.

Figure 3. Font Query Screen

In order to add packages, the installer simply specifies the package location with the Configure button (the packages can come from CDROM or an NFS mounted volume) and then selects the Available button.

The Desktop provides support for associating files with default actions. For example, when double clicked, the Caldera_Info icon (see Figure 4) launches the Arena Web Browser with the contents of the Calder_Info file.

Figure 4. Caldera Network Desktop

The Desktop itself can be configured as a window, as shown in Figure 4, or as the root window, much like MacIntosh's Finder.

Directories can be opened into windows similar to the desktop. For example, if the user wishes to add an icon for **Seyon** to the desktop area, the /usr/X11R6/ bin directory could be opened and the icon for Seyon could be dragged into the Desktop window. This doesn't move the file: it simply creates an icon.

There are many more features offered by the Desktop—covering all of them would make this article far too long—and duplicate the included HTML documentation.

A demo version of the CRISP editor is also included, which is configured as the default desktop action for text files, and as the default text editor. The full version is available to Linux users for a discounted price. It is a very complete editor for programmers and system administrators, from what I can see. (Unfortunately, we can only read about these features, since almost all of them

are disabled in the “lite” version offered for preview, which has fewer features than emacs or vi.)

I was unable to try out the Netware Client features, since the Novell Servers at my office were already “having some problems” (it's a long story). Since Caldera has a working relationship with Novell, I don't think I would have found any problems. Caldera made an effort to integrate Netware support in a way that fits the Linux/Unix file-system paradigm, and from the looks of the documentation, they succeeded. The client locates servers for the user and auto-mounts directory and objects. After being authenticated, the user has automated access to the files for which the account has permissions.

The Caldera Font Server supports TrueType, Postscript and SPEEDO fonts, an easily overlooked feature. Access to these fonts will make software such as WordPerfect even more valuable. It also offers a graphical management tool that allows the administrator to view, install and delete fonts. The BACKUP.UUNET utility is a backup system licensed from MTI, a RAID system manufacturer. This utility alone would be worth devoting a separate review to.

Bundling in a product like this is a key feature of the Network Desktop. This product provides comprehensive network backup and restore capabilities, and makes the idea of supporting an entire office or enterprise system running Caldera Network Desktop a serious possibility.

For Internet connectivity the Network Desktop includes a Web Browser, Web server, compiled versions of **pppd** and **dip**, and the **tcpd** wrapper program. The **tcpd** program provides a very important security feature by allowing the administrator to control who can access workstation services.

In conclusion, the Caldera Network Desktop is a worthwhile product for Linux users of all levels. I read once that the measure of usability for an operating system is this: Which one would you recommend to your computer illiterate friends? Previous to the Network Desktop, recommending a Linux release to these friends meant agreeing to provide unlimited free technical support. With the Network Desktop, Linux becomes in many ways a better bet than Windows 95 or OS/2. For experienced Linux users, the Network Desktop is an excellent bundling of the important network support tools, a beautiful mechanism for installing and removing software, and a handy collection of third party tools.

Figure 5. Sample Directory on CND

As I said earlier, Caldera Network Desktop is more than a single product, and a single article like this cannot provide a fully comprehensive review. I hope,

however, I have provided you with enough information to decide whether or not to use or recommend CND.

Eric Goebelbecker (eric@cnct.com) is a systems analyst for Reuters America, Inc. He supports clients (mostly financial institutions) who use market data retrieval and manipulation APIs in trading rooms and back office operations. In his spare time (about 15 minutes a week...), he reads about philosophy and hacks around with Linux.

Archive Index Issue Table of Contents

Advanced search

Seamless Object-Oriented Software Architecture

Dan Wilder

Issue #22, February 1996

The book spends roughly equal time on notation, process, and case studies.

Authors: Kim Wald and Jean-Marc Nerson

Publisher: Prentice Hall

ISBN: 0-13-031303-3

Reviewer: Dan Wilder

If you have some experience with object-oriented programming, and you are looking for a compatible design method, read this book. The authors describe a method (they avoid the term “methodology,” and in Chapter 6 tell us why) based entirely on object-oriented concepts. In use since 1990, Business Object Notation (BON) avoids starting with the familiar data flow, entity-relationship, or state transition diagrams. System use scenarios, prominent in other methods, are found here, but not in a fundamental role. What you will find are classes, featuring inheritance and client relations; clusters, flexible groupings of classes; and objects, that is, the run-time instances of classes. Original and quite sensible graphical and textual notations are described, suitable for garage floor, white board, or CASE tool. The book spends roughly equal time on notation, process, and case studies. Several appendices present condensed information. A nice glossary and a fine bibliography provide the icing on this cake.

Seamless Object-Oriented Software Architecture is the first widely available full-length discussion of BON. For a fresh look at issues of object-oriented software development, the book is worth reading even if you're happy with another method. If not, consider this one. The book is readable; it doesn't get bogged down in minutiae, but it covers a lot of material. Be warned: these authors hit

the ground running. If you are not already familiar with object-oriented concepts, start with a more introductory book.

Among BON's key ideas are two I will discuss briefly. First, reduce the conceptual gap between design and implementation. Second, provide means to selectively abstract from the welter of low-level details. The two ideas synthesize well. The resulting model is of a single piece, even while a view of it may range over many different abstraction levels. Hence the use of “seamless”. Take a detailed look at a small piece of the model, in a context of the most abstract view of the rest, and it fits into place perfectly.

The conceptual gap between design and implementation is reduced by eliminating difficult, clumsy, or irreversible transformations from the picture. Data flow diagrams, state transition models, entity-relationship diagrams, and so on, while considered useful for specialized problems, are here dismissed as foundations for a general-purpose method. Rather, the effort is to explore the application of class, object, inheritance, polymorphism, and the software contract, to the higher-level representation of systems.

Abstraction is facilitated by the easy transition between levels of detail in the BON models, and also by the rich semantic content lent to the class interface description by the software contract. This contract is a part of the class interface, spelling out the class requirements and obligations, independent of the program code, which often won't exist when the interface is first described. This use of contract provides real substance in the design, in a way that bubbles and arrows just can't do. It does so in a way that is understandable in a context of the more abstract bubbles and arrows. Zoom out for perspective. Zoom in for detail. And the detail always makes sense in the context of the larger picture. Or else it doesn't, and this tells you either the detail or the picture must be changed! Better to find this out early, before the system is nearly implemented, and changes become much more expensive. A good design method should help you find this sort of thing.

The focus is always on the design of coherent, well thought out classes which embody what you know about some concept or idea. These furnish the basis for software re-use. In the short term, within the scope of their originating project, they are fastened together perhaps more than once, as the definition of the project changes, using relatively transient “glue” classes that give a particular system its shape and particulars. Thus re-use begins at home, and the system is not hedged in by premature rigid definition of what is in many cases the most volatile aspect of a system: its external interface.

The notion of coherent re-usable classes bears some kinship to the traditional Unix “small sharp tools” philosophy, where programs that do one thing well

may be combined in unanticipated ways to perform work not contemplated when the tools were written. However, the flexibility of the object-oriented framework is much greater. The key in is having well-focused tools: in the Unix case, binaries like **ls** and **find**; for object-oriented programming, classes like **LINEAR_ITERATOR** or **BINARY_TREE**. Or perhaps **PATIENT_ACCOUNT** or **STEPPER_MOTOR**.

The invention of such classes and their combination with pre-existing classes to form a working system is an incremental process requiring many trips back and forth from high level design through implementation. As in many other methods, you start at a high level, produce a rough cut at a design, then immediately begin implementation. Selected subsystems are targeted, usually not the easiest ones. This provides a reality check on the design. Then, back to the high level to revise by what you have learned, return to implementation, and so on. Implementation throws the cold clear light of day on design, design guides implementation.

From time to time you make a side trip into system use scenarios. These do not direct the organization of the system, but rather test the evolving design. The typical situation: here is something it would be reasonable to do; does this set of classes support the reasonable behavior? Sometimes it doesn't, so you go back and figure out what additional useful ideas might be wrapped in classes. The use scenarios are accompanied by object scenarios, showing the interplay of objects to accomplish the use scenario. A novel graphical notation is used, which allows easy depiction of interactions between many more objects than the conventional ladder or lattice-like interaction diagrams often used elsewhere.

The middle of the book, chapters 6 through 8, discusses the process of system development under BON. Some readers may want to begin reading here, as this part of the book talks a lot more about the “how” and “why” of the method. Nine standard tasks are completed, not necessarily in order, each by some mix of nine standard activities. The tasks, the subject of chapter 7, are:

1. Delineate system borderline
2. List candidate classes
3. Select classes and group into clusters
4. Further define classes
5. Sketch system behaviors
6. Define public features
7. Refine system
8. Generalize
9. Complete and review system

The activities, the subject of chapter 8, are:

1. Finding classes
2. Classifying
3. Clustering
4. Defining class features
5. Selecting and describing object scenarios
6. Working out contracting conditions
7. Assessing reuse
8. Indexing and documenting
9. Evolving the system architecture

Each task and activity is discussed at some length. These authors don't just dump a notation on you and leave you adrift; some care has gone into describing just how you might proceed. While emphasizing over and again that satisfactory performance is not subject to pat answers, but rather requires talent, experience, and insight, Waldán and Nerson nonetheless manage to provide what sounds to me like good advice about each of the tasks and activities. In a literature where solutions that are too simple abound (“Model the physical objects,” “Don't use multiple inheritance,” “Encapsulate interface, data, and process in separate classes”) the thoughtful advice in these chapters is welcome.

I'll be bringing you further report in a few months. With the help of the Linux port of EiffelCase, the BON tool from Interactive Software Engineering of Santa Barbara, California, I will attempt a small freeware project using the advice in this book. My success or failure, and the delights or frustrations encountered, will furnish the topic of my next article.

Dan Wilder (dan@gasboy.com) writes programs and prose in Seattle, Washington. A buildmeister by day, Linux fanatic and newsgroup surfer by night, he also finds time to get outdoors, play with his two darling children, and pick apples.

Archive Index Issue Table of Contents

Advanced search

New Products

LJ Staff

Issue #22, February 1996

ViaCrypt PGP/PE and ViaCrypt PGP/BE, >ParaSoft Releases Insure++ V3.0 and more.

ViaCrypt PGP/PE and ViaCrypt PGP/BE

ViaCrypt has introduced ViaCrypt PGP/PE (Personal Edition) Version 4.0, an updated, easier-to-use version intended to replace its current ViaCrypt Version 2.7.1 message and file privacy software product. ViaCrypt has also announced a new edition of its software with specialized features that specifically address the needs of business users—ViaCrypt PGP/BE (Business Edition) Version 4.0. Both new versions are compatible with all existing PGP versions. Price: ViaCrypt PGP/PE, $129.00 for a single-user license and $390.00 for a five-user license; ViaCrypt PGP/BE, $149.00 for a single-user license and $450.00 for a five-user license.

Contact: ViaCrypt, 9033 N. 24th Avenue, Suite 7, Phoenix, AZ 85021-2847. Phone: 602-944-0773. Fax: 602-943-2601. URL: www.viacrypt.com.

ParaSoft Releases Insure++ V3.0

ParaSoft Corporation, developer of high technology tools for software development, has announced Insure++ V3.0, a new version of ParaSoft's automatic, run-time error detection environment. Version 3.0 enhancements include support for Linux, new thread support, new object coverage analysis, new support for pre-compiled headers, leaks detected by simply linking, reduced memory usage, increased speed, increased ability to detect errors, and dynamic and static linking. Price: $1,995.00 for a single machine license.

Contact: ParaSoft, 2031 S. Myrtle Ave., Monrovia, CA 91016. Phone: 818-305-0041. Fax: 818-309-9048. E-mail: info@parasoft.com. URL: www.parasoft.com.

XRT Widgets Available For Linux

KL Group Inc. has announced that all its XRT widgets, XRT/3d, XRT/field, XRT/graph and XRT/table, are available for the Linux operating system. This move enables developers using Linux to include the XRT family of widgets in their applications. XRT widgets for Linux are about half the standard Unix prices. There are no royalty or run-time fees for distributing end-user applications built with XRT products. Price: $995.00 XRT/graph; $1245.00 XRT/3d; $745.00 XRT/table; $495.00 XRT/field.

For more information about XRT widgets or to get a free 30-day evaluation contact: info@klg.com or visit www.klg.com/.

X11-Based Office Software

Moonlite Information Services has announced its 10 Out Of 1 software, a complete X11-based office software for Linux. 10 Out Of 1 includes a GUI library, a window manager, Littera (a word processor), Anaconda (a spreadsheet application with 2-D and 3-D graphics), Sophia (a database application), many desktop utilities, and the tools needed to develop your own 10 Out Of 1 compliant software. Price: $699.00 commercially, $299.00 for students, educational and research organizations.

Contact: Moonlite Information Services, Steinmatten 2, 79194 Gundelfinden, Germany. Phone: +49 171 3675843.

Archive Index Issue Table of Contents

Advanced search

Upcoming Events

LJ Staff

Issue #22, February 1996

Symposiums and Conferences

Second Symposium on Operating Systems Design and Implementation

Cynthia Deno, the Exhibition and Publicity Coordinator for USENIX at The UNIX and Advanced Computing Systems Technical and Professional Association, writes us that there has been a change in the dates of the symposium from those given in the September 1995 issue of *LJ*.

The Program Committee is currently seeking papers describing original work concerning the design, implementation and use of modern operating systems. Besides mature work, we encourage submissions describing exceptionally promising, well-grounded speculative work, or enlightening negative results. For submission guidelines, please contact the program chairs at osdi@cs.rice.edu.

For more information about the above USENIX events contact USENIX Conference Office, 22672 Lambert Street, Suite 613, Lake Forest, CA USA 92630; phone 714-588-8649; fax 714-588-9706; e-mail conference@usenix.org ; WWW www.usenix.org.

First Conference on Freely Redistributable Software

The First Conference on Freely Redistributable Software (sponsored by the Free Software Foundation) will take place Friday to Monday, February 2-5, 1996 at the Cambridge Center Marriott in Cambridge, MA. Keynote speakers will be Linus Torvalds and Richard Stallman. The conference will feature two days of tutorials on Linux (Phil Hughes), Advanced Emacs and GCC (Richard Stallman) expect (Don Libes), PERL (Tom Christenson), and other topics, as well as refereed papers.

Peter Salus will give seminars entitled ''Linux: An Open System For Everyone'' and “Installing and Running Linux.” The first seminar will look at Linux from its beginnings through its current capabilities, including a look at what some companies are currently doing with Linux. The seminar will conclude with a look at the future of Linux. Peter's second seminar will consist of a ''Look Under the Hood'' covering what makes up a Linux system, what you need, how to install it and what to do when something goes wrong. Interconnectivity options will also be addressed. Requests for registration materials and full programs may be made by e-mail conf96@gnu.ai.mit.edu), phone (617-542-5942) or fax (617-542-2652).

Archive Index Issue Table of Contents

Advanced search

Consultants Directory

> This is a collection of all the consultant listings printed in *LJ* 1996. For listings which changed during that period, we used the version most recently printed. The contact information is left as it was printed, and may be out of date.

ACAY Network Computing Pty Ltd
Australian-based consulting firm specializing in: Turnkey Internet solutions, firewall configuration and administration, Internet connectivity, installation and support for CISCO routers and Linux.

Address:
Suite 4/77 Albert Avenue, Chatswood, NSW, 2067, Australia
+61-2-411-7340, FAX: +61-2-411-7325
sales@acay.com.au
http://www.acay.com.au

Aegis Information Systems, Inc.
Specializing in: System Integration, Installation, Administration, Programming, and Networking on multiple Operating System platforms.

Address:
PO Box 730, Hicksville, New York 11802-0730
800-AEGIS-00, FAX: 800-AIS-1216
info@aegisinfosys.com
http://www.aegisinfosys.com/

American Group Workflow Automation
Certified Microsoft Professional, LanServer, Netware and UnixWare Engineer on staff. Caldera Business Partner, firewalls, pre-configured systems, world-wide travel and/or consulting. MS-Windows with Linux.

Address:
West Coast: PO Box 77551, Seattle, WA 98177-0551
206-363-0459
East Coast: 3422 Old Capitol Trail, Suite 1068, Wilmington, DE 19808-6192
302-996-3204
amergrp@amer-grp.com
http://www.amer-grp.com

Bitbybit Information Systems
Development, consulting, installation, scheduling systems, database interoperability.

Address:
Radex Complex, Kluyverweg 2A, 2629 HT Delft, The Netherlands
+31-(0)-15-2682569, FAX: +31-(0)-15-2682530
info@bitbybit-is.nl

Celestial Systems Design
General Unix consulting, Internet connectivity, Linux, and Caldera Network Desktop sales, installation and support.

Address:
60 Pine Ave W #407, Montréal, Quebec, Canada H2W 1R2
514-282-1218, FAX 514-282-1218
cdsi@consultan.com

CIBER*NET
General Unix/Linux consulting, network connectivity, support, porting and web development.

Address:
Derqui 47, 5501 Godoy Cruz, Mendoza, Argentina
22-2492
afernand@planet.losandes.com.ar

Cosmos Engineering
Linux consulting, installation and system administration. Internet connectivity and WWW programming. Netware and Windows NT integration.

Address:
213-930-2540, FAX: 213-930-1393
76244.2406@compuserv.com

Ian T. Zimmerman
Linux consulting.

Address:
PO Box 13445, Berkeley, CA 94712
510-528-0800-x19
itz@rahul.net

InfoMagic, Inc.
Technical Support; Installation & Setup; Network Configuration; Remote System Administration; Internet Connectivity.

Address:
PO Box 30370, Flagstaff, AZ 86003-0370

602-526-9852, FAX: 602-526-9573
support@infomagic.com

Insync Design
Software engineering in C/C++, project management, scientific programming, virtual teamwork.

Address:
10131 S East Torch Lake Dr, Alden MI 49612
616-331-6688, FAX: 616-331-6608
insync@ix.netcom.com

Internet Systems and Services, Inc.
Linux/Unix large system integration & design, TCP/IP network management, global routing & Internet information services.

Address:
Washington, DC-NY area,
703-222-4243
bass@silkroad.com
http://www.silkroad.com/

Kimbrell Consulting
Product/Project Manager specializing in Unix/Linux/SunOS/Solaris/AIX/ HPUX installation, management, porting/software development including: graphics adaptor device drivers, web server configuration, web page development.

Address:
321 Regatta Ct, Austin, TX 78734
kimbrell@bga.com

Linux Consulting / Lu & Lu
Linux installation, administration, programming, and networking with IBM RS/6000, HP-UX, SunOS, and Linux.

Address:
Houston, TX and Baltimore, MD
713-466-3696, FAX: 713-466-3654
fanlu@informix.com
plu@condor.cs.jhu.edu

Linux Consulting / Scott Barker
Linux installation, system administration, network administration, internet connectivity and technical support.

Address:
Calgary, AB, Canada
403-285-0696, 403-285-1399
sbarker@galileo.cuug.ab.ca

LOD Communications, Inc
Linux, SunOS, Solaris technical support/troubleshooting. System installation, configuration. Internet consulting: installation, configuration for networking hardware/software. WWW server, virtual domain configuration. Unix Security consulting.

Address:
1095 Ocala Road, Tallahassee, FL 32304
800-446-7420
support@lod.com
http://www.lod.com/

Media Consultores
Linux Intranet and Internet solutions, including Web page design and database integration.

Address:
Rua Jose Regio 176-Mindelo, 4480 Cila do Conde, Portugal
351-52-671-591, FAX: 351-52-672-431
http://www.clubenet.com/media/index.html/

Perlin & Associates
General Unix consulting, Internet connectivity, Linux installation, support, porting.

Address:
1902 N 44th St, Seattle, WA 98103
206-634-0186
davep@nanosoft.com

R.J. Matter & Associates
Barcode printing solutions for Linux/UNIX. Royalty-free C source code and binaries for Epson and HP Series II compatible printers.

Address:
PO Box 9042, Highland, IN 46322-9042
219-845-5247
71021.2654@compuserve.com

RTX Services/William Wallace
Tcl/Tk GUI development, real-time, C/C++ software development.

Address:
101 Longmeadow Dr, Coppell, TX 75109
214-462-7237
rtxserv@metronet.com
http://www.metronet.com/~rtserv/

Spano Net Solutions
Network solutions including configuration, WWW, security, remote

system administration, upkeep, planning and general Unix consulting. Reasonable rates, high quality customer service. Free estimates.

Address:
846 E Walnut #268, Grapevine, TX 76051
817-421-4649
jeff@dfw.net

Systems Enhancements Consulting
Free technical support on most Operating Systems; Linux installation; system administration, network administration, remote system administration, internet connectivity, web server configuration and integration solutions.

Address:
PO Box 298, 3128 Walton Blvd, Rochester Hills, MI 48309
810-373-7518, FAX: 818-617-9818
mlhendri@oakland.edu

tummy.com, ltd.
Linux consulting and software development.

Address:
Suite 807, 300 South 16th Street, Omaha NE 68102
402-344-4426, FAX: 402-341-7119
xvscan@tummy.com
http://www.tummy.com/

VirtuMall, Inc.
Full-service interactive and WWW Programming, Consulting, and Development firm. Develops high-end CGI Scripting, Graphic Design, and Interactive features for WWW sites of all needs.

Address:
930 Massachusetts Ave, Cambridge, MA 02139
800-862-5596, 617-497-8006, FAX: 617-492-0486
comments@virtumall.com

William F. Rousseau
Unix/Linux and TCP/IP network consulting, C/C++ programming, web pages, and CGI scripts.

Address:
San Francisco Bay Area
510-455-8008, FAX: 510-455-8008
rousseau@aimnet.com

Zei Software
Experienced senior project managers. Linux/Unix/Critical business software development; C, C++, Motif, Sybase, Internet connectivity.

Address:
2713 Route 23, Newfoundland, NJ 07435
201-208-8800, FAX: 201-208-1888
art@zei.com

Archive Index Issue Table of Contents

Advanced search