Mix 'n' match

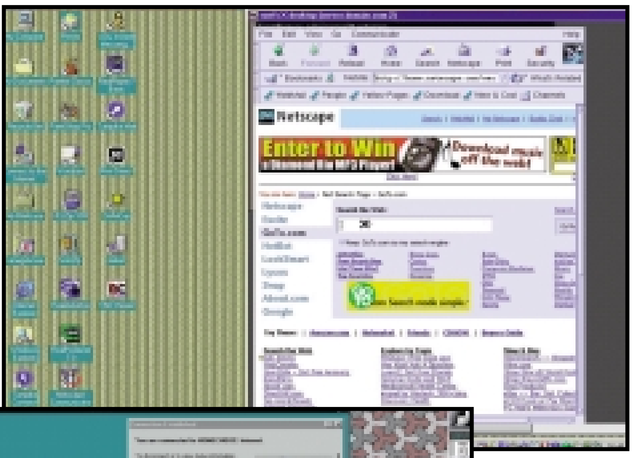
Chris Bidmead tracks down the best methods to let you use a multitude of operating systems.

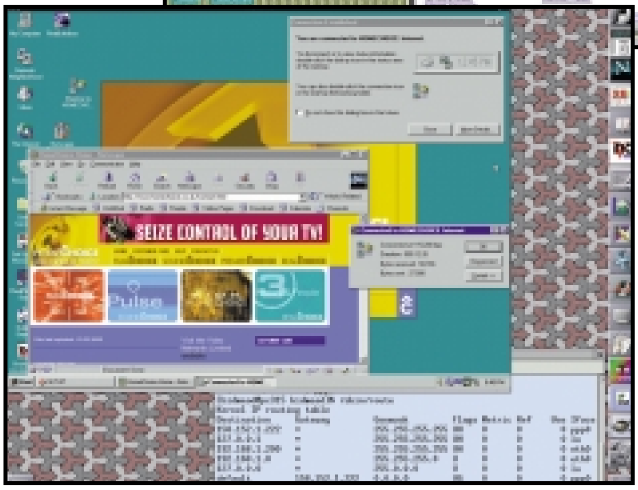
I get a lot of questions from readers who want to run Linux and Windows 'side by side' on the same machine. The traditional way of doing this is by dual booting, but a lot of you are getting more sophisticated.

Toby (me@tobycool.f9.co.uk), writes: 'I would like to use Bochs DOS/Windows emulator in Red Hat 6 – but where is it? In a web search all the pages lead to a page that doesn't exist any more! I can't run Wine for some reason, and I'd rather have an emulator that creates a virtual machine than simply running (MS) programs anyway. VMWare costs an absolute fortune!'

You should never overlook the obvious, as it can be found at http://www.bochs.com. It's also a good idea to check out www.freemware.org.

Strictly speaking, Bochs isn't a Windows emulator, it's an x86 PC emulator that, like VMWare (www.vmware.com), allows you to run one operating system inside another. Bochs is commercially licensed, but FreeMWare (confusingly renamed 'plex86' since I replied to Toby), is an open source development based on it.


This is Peter Rose's Windows desktop, using VNC to display Netscape running on his remote Linux machine...


...I've been doing much the same thing in reverse for the past couple of months. Here Netscape is running on a Windows machine on my LAN and the display pops up on my Linux machine. I did this because HomeChoice, which supplies my ADSL Internet connection, only offers support to a Windows machine. However, with some technical input from the helpful folks at HomeChoice I now have a direct ADSL connection into my Linux box. So this screenshot of VNC displaying a legacy operating system is of historical interest only :-)

If you have two machines networked together, the easiest way is to run one operating system on each and use X or a variant to unite the two displays on a single desktop. We've already looked at ways of doing this with a Windows-based X server like Hummingbird's Exceed (www2.hcl.com/html/forms/nc/exceed/request.html), and we've talked in the past about the free software alternative called VNC (www.uk.research.att.com).

Peter Rose (acsupply@acsupply.demon.co.uk) has come up with an ingenious variant of this idea. He works from a Windows machine connected through a 56K modem to a Linux box that has a fast Internet connection. He was puzzling for some time about the best way to take advantage of this, and VNC gave him the answer.

He writes: 'Well, thanks to your encouragement (and actually going back and reading the VNC docs in more detail:-)) I can now run an X display on my remote Linux box. Early stages yet, but I can now open up Netscape and use it to download the big packages I need on the server at a much faster rate than the 56K dialup I have on my desk. This was my original objective – where many packages now have to be downloaded via a browser and direct command-line ftp is not available, it was rather tedious to have to spend two hours on a large download to my local machine and then have to repeat the process to send it back to my remote server.'

Back to backup

I harp on about this one because as a 20-year veteran PC user I've lost a ton of work over that time, and I don't want this to happen to you. There are some subtle dimensions to backup that you only come to appreciate over the years. One of these – particularly applicable to the adventurous folks who read columns like this one – is the way lack of backup tends to stick you with out-dated executables that you daren't upgrade for fear of disturbing the rest of your working system. Backup in this context is a safeguard that opens up the possibilities of exciting adventures into safe computing!

The other piece of wisdom that only emerges over time is that backup (temporary assurance against data loss) and archiving (long-term storage of data) tend to converge. I find myself returning to backup media years later trying to recover files that may have seemed trivial at the time, but have since become important. The trouble is, some of those backups were made on obsolete devices, using proprietary backup packages from long forgotten software companies, running on a no longer supported operating system.

There's not a lot you can do about the obsolescence of the backup devices themselves. All through the Nineties sticking to the well-established 4mm DAT standard seemed to be the answer. New DAT devices with higher capacities came out every few years, but they were backward compatible. These days my Hewlett-Packard DDS3 SureStore DAT24 manages to compress an impressive 24GB onto a small cartridge, and can still read my old 2GB DDS1 tapes. There's a higher capacity DDS4 out now, and DDS5 is promised. But many fear the format is going to run out of leg room over the next few years, and at some stage I'm probably going to have to transfer my key archives across to something that uses physically bigger cartridges.

One candidate is a new 8mm format from Ecrix (http://www.ecrix.com), and happily I've managed to get hold of one of its VXA tape drives for evaluation. I like the compressed capacity of 66GB per cartridge, and I'm impressed by the reliability claims: apparently you can freeze a cartridge in a block of ice, or drop it into boiling coffee, and still get your data back! (see the website for the full sensational details). Circus tricks apart, what I like about this drive is that it's an excellent fit with my current ideas about backup software. It doesn't come with a particular proprietary package that you must use, or with a driver that ties you to a particular operating system. It's a generic SCSI device that works with standard Unix backup software.

I think it's a good rule to avoid a proprietary backup package, however convenient it may seem at the time. It will probably write your data in a unique format that other software will find difficult if not impossible to recognise. And I also like to avoid software that ties me into a particular operating system – even if that OS is Linux.

There's no perfect solution to this, but for the past few years it has seemed sensible to me to use GNU tar for all my really important backups. Because it's available as source, it's highly portable across operating systems, and is pretty well guaranteed not to disappear overnight (or even over a few decades).

As a standard SCSI device, the Ecrix VXA tape drive drops straight in as a substitute for the HP SureStore DAT24. In fact I run them side by side on the same SCSI bus, where they appear to Linux as /dev/nst1 and /dev/nst0 respectively.

GNU tar is a command line utility, and its incantations can be very convoluted. And if you're going to write multiple sessions to the same tape (and with 66GB per tape that makes a lot of sense) you'll also need the mt utility to position the tape. It is standard Unix procedure to roll your own script to do this. One of my backup scripts can be seen in Fig 1.

GNU tar is a command line utility and its incantations can be very convoluted

FIG 1

Backup script

```
#! /bin/bash
# incremental backup of $DIR
# chb 12 Jan 00
# To make a master backup, just delete the snapshot file
# NB, the exclude file must NOT include any blank lines -
# not even a final empty line.
# implement the curly bracket wrapper below if you want to ↙
log
# all this (and see nothing directly on the screen).
#{
[ $# -eq 0 ] && exit 0
DIR=$1
BUTYPE=Incr
# if $DIR/tar.exclude doesn't exist we need to make it, else
# tar will refuse to run.  Hmmm... assumes we have write ↙
permission here
[ -e $DIR/tar.exclude ] || touch $DIR/tar.exclude
# if this file exists - and is NOT EMPTY (-s) - then its a ↙
Main b/u
[ -s $DIR/tar.snapshot ] || BUTYPE=Main
mt seod
tar cv \
-V "`date` $BUTYPE ($DIR)" \
-X $DIR/tar.exclude \
-g $DIR/tar.snapshot \
$DIR
# } >> ~/backup.log 2>&1
```

(*Key: ↙ code string continues*)

This script introduces another dependency – on the bash shell. But bash is another GNU free software utility, and is no less portable than tar.

I call this script incr and will normally evoke it as root, adding the name of the directory to backup as the command line parameter. When GNU tar is evoked with the -g switch it manages a tracking file (here named tar.snapshot) that keeps a record of the current state of the directory, so that at the next tar -g session it will only back up changed files. To do a non-incremental backup of the whole directory it's only necessary to delete tar.snapshot. The script will label the tarball 'Main' or 'Incr' depending on which you choose. The way I've set it up here implies that whoever runs incr has write access to the directory; but a better way might be to keep the snapshot files in a special incr-owned directory.

You'll notice mt being used to kick off the session by positioning the tape at the end of data with the 'seek end of data' command, or seod. With the GNU implementation of mt you could equally well use eod, which is a synonym for seod. But I often use mt directly from the command line and there's a frighteningly

FIG 2

Tapelist script

```
#! /bin/bash
# wind through the tape stopping at each filemark
# to examine the next block, hopefully an archive
# label or a meaningful initial tarball entry

# New version chb 12 Jan 00 can pick up at any block
# supplied as a criterion on the command line.
# Useful for long tapes.
# Minor mod now puts <<eod>> in line with EOD block number
#
# Modded again 9 Feb 00 to remove rewind at end, and
# allow the list to start from the current block (tl x)

TAPE=/dev/tape ; export TAPE
INPUT=$TAPE
EOD="<<eod>>"
BLOCK='mt tell | colrm 1 9 | tr -d "."'
DATA='dd if=$INPUT count=1 2> /dev/null'
if [ $# -eq 0 ] ; then  LOCATION=0
else LOCATION=$1
fi
if [ "$LOCATION" = "x" ]
then LOCATION=`eval $BLOCK`
fi
echo "Starting at Block $LOCATION"
mt seek $LOCATION && {
[ $LOCATION -eq 0 ] || mt fsf 2>/dev/null
printf "%10s  %s" "`eval $BLOCK`" "`eval $DATA`"
while mt fsf 2> /dev/null ; do
printf "\n"
printf "%10s  %s" "`eval $BLOCK`" "`eval $DATA`"
done
printf "${EOD}\n"
}
```

FIG 3

Tape script

```
#!/bin/bash
# 24 Feb 00; chb
# script to indicate current ↙
/dev/tape linkage or change ↙
it to
# one of two different SCSI ↙
tape devices at /dev/nst0|1
# currently requires DAT at ↙
/dev/nst0 and VXA at /dev/nst1
# but should probably use ↙
variables to be more easily ↙
modified
if [ $# -gt 2 ]
then
echo "Usage is $0 DAT or $0 ↙
VXA"
exit -1
fi
case $1 in
DAT|dat )
# echo "OK, DAT"
ln -sf /dev/nst0 /dev/tape ;;
VXA|vxa )
# echo "OK, VXA"
ln -sf /dev/nst1 /dev/tape ;;
esac
link=`ls -l /dev/tape | ↙
colrm 1 74`
case $link in
nst0 )
echo "Tape is DAT" ;;
nst1 )
echo "Tape is VXA" ;;
esac
```

(*Key:* ↙ *code string continues*)

similar mt command called eof (also known as weof), which will 'write end of file' at the current point on the tape. As I've discovered, this can be disastrous, so I've promised myself never to use eof or eod, sticking instead to seod and weof.

When I use GNU tar to back up direct from the command line I usually do something like this:

```
mt seod
tar cvV "`date` Prior to ↙
upgrading to kernel 2.2.13" ↙
/usr/src
```

...which writes a useful identifying label onto the tarball. To display these labels I wrote a script some years ago called tapelist. I've improved it since I first published it back in March of last year, and the latest version is in Fig 2.

One of the changes I've made (which probably should have been documented in the script) is that, as you can see, it's permanently targeted at a device called /dev/tape. This is just a symlink to one of the real tape devices, either /dev/nst0 or /dev/nst1. As long as I make sure I have a permanent environmental variable $TAPE with the value '/dev/tape', both mt and tar will default to acting on the same device. A separate script, just called tape, manages the link between /dev/tape and the required real device (see Fig 3).

Just typing tape at the command link will indicate the present linkage, or this can be changed by typing tape followed by DAT or VXA.

These three tape scripts aren't particularly sophisticated – in particular you'll notice that I'm not running them under cron (a very simple next step) and there's no automated way of getting the data back. What I do to restore the data is look at a filed or printed copy of the relevant tape list, manually position the tape with mt, and then tar -x from there.

I deliberately haven't set out to design an elaborate all-singing, all-dancing automated backup regime, because I've been stung by such things before. They second guess your needs, which often change from day to day, and they may give you a false sense of security, because behind the fancy front end you probably haven't a clue what the software is up to. My purpose here is to understand at a fairly low level what the tape device is doing, and what I'm doing with it. I do intend to develop further automation around this in time – but I'm in no hurry.

CONTACTS

Chris Bidmead welcomes your comments on the Unix column. Contact him via the *PCW* editorial office or email: **unix@pcw.co.uk**