Setting up a Psychology Research Lab with Linux

David W. Allbritton

DePaul University

Address correspondence to:

David Allbritton
Department of Psychology
DePaul University
2219 N. Kenmore Ave
Chicago, IL 60614

dallbrit@depaul.edu
(773) 325-4799

Abstract

The Linux operating system is described, and features that make it a useful tool for inexpensively setting up a lab for psychological research are identified. Among the features of Linux that are identified as useful to researchers are its low cost, stability, configurability, ability to run on older hardware, and features for managing multiple users and operating systems. The use of Linux in the author's lab is described as an example, with particular attention to the steps for installing Linux, setting up multi-booting and networking with Linux. Additional sources of information about Linux installation, configuration, and available programs are also identified.

Setting up a Psychology Research Lab with Linux

Setting up a research lab can be a daunting task, particularly when working within a limited budget. Typical needs would first include multiple workstations equipped with software for conducting experiments. In addition to running experiments, the same machines are likely to be used as workstations for other tasks such as experiment preparation, data analysis, and document preparation. Most researchers are accustomed to using software running under some variant of the Microsoft Windows operating system for many of these tasks. The software for running the experiments, on the other hand, may be an MSDOS program. Many researchers choose to use DOS for running experiments because the absence of multi-tasking makes it easier to achieve millisecond timing (see MacInnes & Taylor, 2001, however, for a description of how millisecond timing can be achieved in multitasking operating systems). Thus many researchers will find that they need to have two different operating systems installed on a single machine. In addition, some system for transferring experimental materials to each workstation and backing up data from each workstation is needed. Although copying files by hand via floppy disk may do in a pinch, what is really needed is a way to automate data backups over a network.

This article describes one solution that relies on Linux, a freely available version of the Unix operating system. Although in my lab I primarily use an MSDOS program, Cogsys, for running experiments, it is quite possible to use Linux for data collection with millisecond accuracy (MacInnes & Taylor, 2001). In fact, descriptions of Linux programs for data collection have been presented in previous issues of this journal (Finney, 2001a, 2001b). Many researchers who would like to use programs such as these may be unable to do so simply because they are unfamiliar with Linux and are not sure how to get started. I hope here to provide both an explanation of why Linux is well-suited to the needs of research psychologists and a guide for getting started with Linux and finding further information.

I will first briefly introduce the Linux operating system and outline some of its advantages for setting up a low-cost experimental laboratory. I will then describe the way that I have used Linux in my lab as one example of how Linux can be of benefit to researchers.

What is Linux?

Linux is an implementation of the UNIX operating system that runs on PCs. Originally developed for the 386 family of Intel processors, Linux is now available for multiple platforms including all x86-compatible processors (i386 or later) as well as Alpha, SPARK, UltraSPARK, StrongARM, PowerPC, and Motorola 68x0 (MacIntosh). Linux runs on machines as big as the IBM 390 series, as small as the Palm Pilot, and even on some versions of the Sony Playstation 2 (http://news.zdnet.co.uk/story/0,,s2085867,00.html). The core of Linux, the kernel, was originally developed by a Finnish graduate student, Linus Torvalds, for whom Linux is named. Torvalds' kernel was modeled on the UNIX operating system, developed in the early 1970s by researchers at Bell Laboratories led by Ken Thompson. The full Linux operating system was subsequently developed by hordes of volunteer programmers from around the world working together over the internet. More recently Linux has been supported by major players in the computer industry such as IBM, which now sells servers with Linux pre-installed.

A full Linux system is generally bundled as a *distribution*, including not only the operating system itself but also many applications and utilities to create a full working environment. There are many distributions of Linux available, providing many different "flavors" of Linux to choose from. Although most Linux applications generally work with any distribution, different distributions vary in what software they include, how software packages are installed and managed, and whether they emphasize security and stability, ease of use, or other specific needs of a particular type of user. There are currently two major graphical desktop environments available for Linux, KDE (www.kde.org) and Gnome (www.gnome.org), both of which are included in most distributions and should be easy for Windows users to navigate. Also included in most distributions are free programs for image manipulation (www.gimp.org), burning music and data CDs (www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html), and web browsing (browsers.netscape.com, www.konqueror.org, www.mozilla.org).

Although the free emulator Wine (www.winehq.com) allows Linux to run some Windows programs, most Windows applications can not run under Linux. In particular Microsoft Office is not available for Linux – which is probably one of the chief reasons that Linux has not yet made serious inroads into the desktop operating system market. There are, however, both free and commercial office applications available for Linux. The two most complete office suites for Linux are Star Office 5.2 from Sun Microsystems (www.sun.com/staroffice), and Applixware 5.0 from Vistasource (www.vistasource.com). Both include full-featured components for word processing, spreadsheets, and presentations. Star Office is free, and can import most Microsoft Office files (including Powerpoint presentations). Applixware sells for about $40 to $100 and has the best import functions for MSWord documents of any Linux word processor.

There are several features of Linux that make it an attractive option for setting up a lab on a budget. First, Linux is free. It can be downloaded free of charge over the internet, including all of its source code as well as executable binaries that have been compiled for a particular processor (See www.linux.org or www.linux.com for a list of available distributions and download sites, as well as useful information for Linux beginners). For those who lack either the equipment or the desire to download and burn their own Linux installation CDs, CDs can be purchased for as little as $5 from various re-sellers, or for about $50-$100 bundled with extra software, manuals, support, and other niceties from vendors such as Redhat (www.redhat.com), Corel (linux.corel.com), Caldera (www.caldera.com), and SUSE (www.suse.com). Linux is also free in the sense of "free to do whatever you want" with it. The Linux kernel and core components are released under the terms of the GNU General Public License (www.gnu.org), which allows anyone to use and modify the software any way they like, so long as the source code for any modifications are then made publicly available under the GPL. This is important to experimental psychologists who may wish to develop programs that build on the functionality of existing GPL programs.

Second, Linux brings the power and stability of the Unix operating system to the desktop PC. In the ancient past (10 years ago) Unix was found only on expensive workstations and mainframe computers. With Linux, an inexpensive desktop PC can now become a fully functional Unix workstation complete with a graphical user interface based on the X-Windows System. Even an older machine that lacks the memory or processing speed to run current versions of Windows software can, with Linux, be revived to host a web site using the free Apache web server (www.apache.org), or to act as a file and print server for Windows machines by running SAMBA (www.samba.org). I have successfully used 486 PCs with as little as 16 megabytes of RAM as dual-boot machines with DOS software for data collection and Linux for automated data backups and software updates. Even a 386 or 486 with no hard drive can, with a specialized "mini" Linux system on a single floppy disk such as Freesco (www.freesco.org), find new life as a router and dialup server for a local network.

Third, Linux makes it easy to administer accounts for multiple users. Like other Unixes, Linux differentiates ordinary users from the system administrator (or root account), and each user is required to enter a username and password to access the system. Each file and process is owned by only one user, and ordinary users do not have access to critical system files or processes. Each file also has a set of permissions associated with it that determine whether any ordinary user besides the file's owner can read or modify that file. By setting file permissions and defining groups appropriately, one can easily control user access to private and shared files. This is a very useful capability for experimental psychologists who may have a number of research assistants, graduate students, and undergraduates using the same lab machines for several different purposes. With Linux's system of user accounts and file permissions, potential disasters such as one user accidentally deleting another user's critical files can be easily prevented.

Fourth, Linux allows you to maintain multiple operating systems on a single machine. By dividing a hard drive into multiple sections or *partitions*, it is possible to have Linux installed on one partition, Windows on a second partition, and DOS on a third on the same machine. Two *bootloaders* are available for Linux to allow the user to choose which operating system to boot when starting the machine. The first, LILO, has been around the longest and is the most widely used. The newer GRUB bootloader is more flexible than LILO and is likely to become the new standard in the future[1]. Although Linux uses a different type of filesystem, Linux can read and write to DOS and Windows partitions. With the commercial software VMware (www.vmware.com) it is even possible to run Windows and Linux at the same time on a single computer, with Windows running in a "virtual machine" under Linux.

Finally, Linux provides reliable and secure means for remotely managing lab workstations. If your lab is on a secure network (such as a local network that is not connected to the internet) then telnet, ftp, and rsh can be used. Telnet allows remote logins to your Linux system over the network, ftp can be used to transfer files to and from your lab workstations, and rsh (remote shell) permits you to execute commands remotely without having to provide a password. Though convenient, all three of these tools are insecure because they transmit unencrypted passwords over the network where they could potentially be intercepted (telnet and ftp) or because they could allow unauthorized users to gain access fairly easily (rsh). A more secure alternative to telnet and rsh is secure shell (ssh), available in both free (openssh) and commercial versions. Combined with the file transfer tool rsync, ssh provides a convenient and secure means for transferring and updating data collection programs and backing up data. Once ssh is properly configured, cron can be used to execute the commands to perform data backups automatically at a certain time each day. Linux lets you control exactly which network services will be available on your machines, making it easier to close potential security holes while configuring all of the functions that are needed for getting your research done.

In short, Linux allows the researcher to have full control over the equipment upon which his or her research depends. This means that you have the flexibility to shape your lab to fit the needs of your research. It also means that you can more easily customize your network connections to fit your needs. Most importantly, it means that you will never find your research program on hold while you wait for someone else to install and configure proprietary operating systems or networking software.

Setting up a Lab with Linux: An Example

In the remainder of the paper I will describe the way that Linux facilitates work in my lab, including specific set-up information and references to relevant resources where appropriate. In my lab, most of the data is collected using Cogsys, a real-time experiment system that runs under DOS on i386 compatible PCs (see Ratcliff, Pino, & Burns 1986, and http://wally.psych.nwu.edu/compute/site/cogsys/cogref_html/cogref.html). Unix utilities such as Awk and Sed, together with shell scripts and Perl programs are used in Linux to prepare and counterbalance (primarily textual) experimental materials and to perform basic data formatting and analysis after data collection. Linux shell scripts also perform automatic backups of the data to a Linux server, where the data is then archived to CDR media. Most of the lab machines also are occasionally used to run programs under Windows 98, the operating system that was pre-installed when they were originally purchased.

Equipment

Four PCs are configured for use as workstations in the lab. Three are Gateway E3200 desktops with Pentium III 350 megahertz processors, 128 megabytes of RAM, and 6 gigabyte IDE hard drives. The fourth workstation has a 180 megahertz Pentium processor, 32 megabytes of RAM and two IDE hard drives (2.4 G and 530 M). Each machine has an SMC Ethernet card, a 3.5 inch floppy drive, and all but one are equipped with a CDROM drive.

A fourth Gateway E3200 in another office serves as a general-purpose Linux workstation and is used to back up data from the lab workstations. It has a 40 gigabyte hard drive and has a Backpack 192100 CDRW drive connected to the parallel port. It is otherwise identical to the three Gateway PCs in the lab.

Getting Linux

RedHat Linux 7.1 is installed on each of the lab workstations. I have chosen to use RedHat Linux because of its ease of installation, personal familiarity (I have used various versions of RedHat Linux since 1995), and large user base. A large user base is helpful because it increases the likelihood that any problems one might encounter have already been solved by some other Linux user and the solution posted on either a web page or discussion group where they can be located through a web search engine like Google (www.google.com). The procedures described here will therefore be specific to RedHat Linux in some respects. Any other modern Linux distribution could serve the same functions, however, and new Linux users should consult linux.com or www.linux.org for helpful descriptions of available distributions.

Linux is usually installed from one or more CDROMs. There are three main ways to get RedHat Linux installation CDs. First, they can be purchased along with a book about Linux from any bookstore that has a computer section. Second, they can be purchased online, either directly from RedHat or from any number of other sources. Because RedHat's license allows anyone to copy and distribute the software, CDs identical to those sold by RedHat are often sold by other vendors, sometimes for as little as $5 each. Purchasing directly from RedHat, on the other hand, entitles the user to technical support for a limited time period and additional documentation and application CDs. Third, the CD disk images can be downloaded over the internet to create your own installation CDs. Simply use either a web browser or ftp to download the files seawolf-i386-disc1.iso and seawolf-i386-disc2.iso from ftp://ftp.redhat.com/pub/redhat/linux/7.1/en/iso/i386/ or from a RedHat mirror site (see www.redhat.com/download/mirror.html) and then use any CD creation software to burn the CDs.

Installing Linux

Partitioning. The biggest challenge facing the new user installing Linux for the first time is re-partitioning the hard drive. Each operating system typically resides on its own separate section of the hard drive, or partition. On most desktop computers running Windows, the entire hard drive is allocated to a single partition, leaving no room for installing a second operating system. Disk utilities such as fdisk can be used to delete the existing partition and create two or more smaller ones in its place, but doing so destroys all files and programs on the existing partition. If you need to keep your existing Windows system intact, or simply are not comfortable with the idea of wiping out everything on your computer and starting from scratch, there are some alternatives.

First, the hard drive could be left as it is, and Linux could be installed within a portion of the existing Windows partition. RedHat 7.1 refers to this option as "partitionless installation," and a number of other distributions specifically designed to be "Windows-friendly" also offer this method. This installation method allows one to try out Linux without making a full commitment. It has significant disadvantages, however. A Linux system installed in this way will suffer slower performance than if installed in its own partition using the native Linux disk format (ext2 or ext3). It will also be subject to a greater risk of data loss, since anything (such as a virus) that damages the Windows file system could also damage the Linux files residing within it. Installing within a Windows partition also forfeits ones ability to use the Linux boot loader to manage multiple operating systems. For these reasons, I do not recommend this solution to the partitioning problem.

The simplest solution is to install Linux on a second hard drive, leaving the original hard drive and Windows installation intact. With 20 gigabyte IDE hard drives currently available for $70 or less, this is a good option. In keeping with the theme of a lab on a budget, however, I will assume that each machine has only one hard drive on which to run Linux, Windows, and possibly DOS. Thus the third option, resizing the existing partition, is the one that will be described here.

Once Linux is installed, the Linux utility **parted** can be used to resize partitions without destroying existing data, but that does not help with the problem of creating a new partition on which to install Linux in the first place. Assuming you want to install Linux on a hard drive that is currently occupied by Windows, there are two options. Commercial software such as Partition Magic (www.powerquest.com) and Partition Commander (www.v-com.com) allow a partition to be resized from within Windows while it is in use, and both are reported to work well. Alternatively, two DOS programs are available for resizing partitions: FIPS (www.igd.fhg.de/~aschaefe/fips/), which I have used successfully in the past, and Partition Resizer (www.zeleps.com), which I have not tried. Both are free, but somewhat more difficult to use than the commercial packages.

Before doing any partition resizing, back up any important files. It is also a good idea to create a Windows boot disk and copy the programs *format* and *fdisk* to a floppy disk just in case. Then follow the instructions accompanying the partitioning software to reduce the size of the existing Windows partition by the amount of space that will be needed to install Linux and any other additional operating system (such as DOS). A RedHat 7.1 installation requires from 300 to 2400 megabytes, depending on what packages are included. For a full workstation installation including a graphical interface, about 2.5 gigabytes is recommended (1.2 minimum). For a server-only installation without a graphical user interface 500 to 600 megabytes may be sufficient.

<u>Running the Linux Installation Program</u>. If your computer's BIOS supports booting from a CD, the installation CDs will be all you need – simply place the CD in the drive and reboot. If not, you will need to create an *installation boot disk* to start the installation process. The installation CD has a directory called "images" containing disk images for creating a boot disk,

and a DOS/Windows program called “rawrite.exe” that can be used to write the image to a floppy disk. After creating the boot floppy, insert it and the first installation CD and reboot. The RedHat installation program will then guide you through the rest of the process, including creating a root (/) partition on which to install Linux, and setting up networking – both of which warrant some advance planning.

When partitioning, remember to leave enough free space (not allocated to any partition) to install any additional operating systems. It is best to use extended partitions (numbered 5 and above) for your Linux installation rather than primary partitions (numbered 1 to 4), and to place the Linux partitions at the end of the hard drive (higher numbered cylinders) rather than the beginning (lower numbered cylinders) if you will be installing DOS or a second version of Windows. The reason is that DOS will only boot from the first primary partition on the hard drive, while Linux does not care where its root partition is located (generally speaking). To control how much disk space is allocated to the Linux partition, you must select the option “Manually partition using Disk Druid” during the installation procedure.

Setting up networking requires that the user supply some information if the machine will have a static IP address (rather than one assigned by a DHCP server somewhere on the network). If you use a static IP address, you will need to get the following information from your system administrator before you begin:

- The IP address and machine name of your workstation
- The netmask
- The IP address of the default gateway
- The IP address of the nameserver

During the installation, you will be asked whether you want to create a *system boot disk* and whether you want to install LILO, the Linux boot loader. Answer yes to both, and select the option to install LILO on the master boot record (MBR) of the first hard drive. For the rest of the installation process, the default selections should work well for most users.

Dual Booting with Linux

When the computer is rebooted after the Linux installation, the LILO boot prompt should appear, offering a choice of booting the original Windows system or Linux. The RedHat installation program thus makes it quite easy to set up a workstation for booting two operating systems. In my lab, however, the workstations need to be able to boot three operating systems - Linux, Windows, and DOS – and that can be a bit trickier. The problem is that DOS needs to be booted from the first partition on the first hard disk, and Windows also demands to be installed on the first partition on the hard drive. Also, if one attempts to install DOS on a hard drive that contains a Windows partition, DOS may overwrite that partition during installation, wiping out Windows. The following is one solution to this problem that has worked in my lab, described step by step.

After installing RedHat 7.1 as described above on a workstation with a single 6.1gigabyte hard drive, the first primary partition (referred to as /dev/hda1 by Linux) was 3.3 gigabytes and contained the machine's original Windows system. The first two extended partitions (/dev/hda5, 200 M, and /dev/hda6, 2.4 G) contained the Linux swap partition (for virtual memory) and the Linux operating system, respectively. The remaining 700 M of the hard drive was left unassigned (not part of any partition) for future use.

Because DOS must be installed at the beginning of the hard drive, the first step was to resize and move the Windows partition to make room for a DOS partition. After booting Linux, log in as root and open a terminal window or console. At the command line prompt, issue the command **parted** to invoke the Linux partition editor[2]. To display the current information, type **print**. Table 1 displays an example of the information reported by **parted** for one of my lab machines.

Insert Table 1 about here

Next use parted's **resize** command to free up space for the DOS partition. For this machine, the following command created approximately 250 M of free space for DOS:

resize 1 251 3255.358

The hard drive then had the first 250 MB unallocated to any partition, the next 3 G allocated to /dev/hda1 on which Windows was installed, and the remaining 3.3 G allocated to the extended partition /dev/hda2. After resizing partitions, be sure to reboot the machine before continuing.

The next step was to install DOS in the 250 M of unused space at the beginning of the hard drive. Before doing so, however, the partition containing Windows (/dev/hda1) had to be "hidden" to prevent DOS from installing itself there instead. This was accomplished by changing the partition type of /dev/hda1 to something DOS would not recognize (and would therefore ignore). To hide the Windows partition, enter the command **fdisk /dev/hda** (again logged in as root). Then press **t** and change the Windows partition (**1**) to type **80**. Press **a 1** to turn off the bootable flag for the Windows partition. Then press **q** to exit from **fdisk**. Table 2 displays the output of **fdisk** for my machine after changing the partition type.

Insert Table 2 about here

The machine should now be ready to have DOS installed. Reboot from a DOS installation floppy disk, and then use the DOS **fdisk** program to create a primary active DOS partition. (Note that although the DOS partition is physically at the beginning of the hard drive, Linux will identify it as /dev/hda3 because it was the third primary partition created.) Next a DOS filesystem should be created using the DOS command **format c: /s**. At this point Cogsys could be installed from floppy disks, or the files could be transferred later after rebooting to Linux.

With each of the three operating systems now installed on its own partition, the final step in setting up the machine for multi-booting was to configure LILO to boot each of them. Edit the file /etc/lilo.conf to include entries for all three operating systems (see Table 3, an example lilo.conf file from my lab machine, for details.) Then issue the command **lilo** to install the new configuration. The next time the machine is restarted, LILO will offer Linux, Windows, and DOS as booting options.

Insert Table 3 about here

Accessing DOS and Windows Filesystems from Linux

Because DOS and Windows files can be accessed under Linux, I use Linux to transfer experimental materials to the DOS partitions of my lab workstations and back up data and other files from the DOS and Windows partitions over the network. The first step for accessing DOS/Windows files in Linux is to create a *mount point* for each of the partitions – an empty directory where the files from that partition will be accessed. I create the mount points /dos-cogsys and /win for DOS and Windows respectively using the following two commands:

mkdir /dos-cogsys

mkdir /win

The partitions can then be mounted manually using the following commands:

```
mount –t vfat /dev/hda1 /win
mount –t vfat /dev/hda3 /dos-cogsys
```

To have both partitions mounted automatically each time Linux is started, add the following two lines to the file /etc/fstab:

```
/dev/hda1       /win           auto      defaults                 2  2
/dev/hda3       /dos-cogsys    auto      defaults,owner,user      2  2
```

The first line causes the Windows C: drive to be mounted in the directory /win, with the files being read-only for regular users and giving write permission only to root. The second line mounts the DOS partition on /dos-cogsys the same way, but allows any *user* to unmount (**umount**) and mount the filesystem, and gives read-write access to the user who mounts it (by making that user the *owner* of /dos-cogsys). Thus any user can copy experimental materials to the DOS partition for use in Cogsys, but only root can make changes to Windows files.

Linux Tools and Utilities

For counterbalancing and randomizing stimulus materials, checking test lists for errors, and preparing data for analysis, I have found Linux's standard UNIX utility programs (such as **sed**, **awk**, **sort**, and **uniq**) to be indispensable. One of the chief advantages of using these utilities is that once a particular problem has been solved, the steps for the solution can be recorded in a *shell script* and re-used the next time the same problem is encountered. The *shell* is the program that the user interacts with when typing instructions at the command prompt. Any series of commands that can be given from the command prompt can also be executed from a file, or script. Because the standard Linux shells (bash and tcsh) include basic programming functions such as looping and conditional execution, just about any task needed for stimulus preparation and data formatting can be accomplished with a shell script. Although a Perl program may be a better choice for complex text and data formatting tasks, for many problems shell scripts are a quick and easy alternative. For example, I have shell scripts that I frequently use to merge two files based on a matching field, calculate means by subject and condition from raw data, and even perform basic statistical analyses.[3]

In addition, a number of free applications are available for Linux that are likely to be useful to experimental psychologists. For statistical computing and graphing, there is the *R* project (www.r-project.org), a free implementation of S-Plus. Gimp (www.gimp.org) provides image manipulation capabilities similar to those of Adobe Photoshop. LaTeX (www.latex-project.org) provides document preparation and typesetting capabilities that are particularly useful for handling mathematical symbols and formulas. Apache (www.apache.org) makes it possible to set up and configure your own web server. These are just a few of the programs freely available for Linux – many more are indexed at sites such as www.rpmfind.net.

Networking with Linux

Connecting a Linux workstation to an existing network is straightforward – the only configuration required is usually guided by the Linux installation program, as described previously. If, for some reason, you need to set up your own local network instead, a Linux workstation can also be configured to serve as a router, nameserver, and dhcp server if needed – although one would be well advised to study the relevant HOWTOs before embarking on such a project.

Once all of the workstations are connected to the network, Linux provides a number of tools that are useful for remote management, transferring files, and automating backups. I use

rsync and **ssh** for file transfers and remote logins, and **cron** to schedule nightly backups. The cron daemon, which carries out tasks at scheduled times, is installed by default. The packages rsync, openssh, openssh-clients, and openssh-server can be installed from the RPMS directory on the RedHat CD (see the man page for **rpm** for more information on installing packages).

Openssh, the free implementation of secure-shell or ssh, is a more secure alternative to telnet, rsh, and ftp because, unlike those three services, ssh does not send either passwords or data over the network unencrypted. Data is first encrypted using a *public key* before being sent over the network, and then decrypted using a corresponding *private key* known only to the receiver. The **ssh-keygen** command generates a user's public and private keys. To allow remote execution or login without a password (for automated backups across the network, for example) the remote user's public key should be placed in the file **.ssh/authorized_keys** in the local user's home directory. By combining **ssh** and **rsync**, files can be securely transferred from one workstation to another[4]. For example, the executing the following command from workstation1 copies the cogsys directory from workstation1 to workstation2:

```
rsync –e ssh –av /dos-cogsys/cogsys workstation2:/dos-cogsys
```

(See the rsync man page for further details.) If the user's public key from workstation1 is listed in the .ssh/authorized_keys file of the same user on workstation2, the files are copied without further user input. If not, the user would be prompted to enter his or her password before the files are copied. Adding the following line to the user's cron file on workstation1 (see the man pages for **cron** and **crontab**) backs up the data from the directory /dos-cogsys/cogsys/output on workstation2 to the directory /home/backups on workstation1 each night at 1 a.m.:

```
0 1 * * *      rsync –e ssh –av workstation2:/dos-cogsys/cogsys/output/ /home/backups
```

By following the same procedure for each lab workstation, all of the data collected each day can be automatically backed up on a single computer, where it can then easily be stored on tape (using **tar**) or CD (using **mkisofs** and **cdrecord**).

What Next? Getting Help with Linux

I have outlined steps that I hope will help researchers who are new to Linux get started, but undoubtedly the new user will quickly discover other tasks to tackle with Linux. At that point the critical question is where to turn for help. Besides paid support contracts from Linux vendors, numerous free sources of help are available for Linux. A good place to start is with **man** and **info** pages on your Linux workstation. Extensive usage notes and documentation are available for most Linux commands by typing **man command-name** or **info command-name** on the command line. Another indispensable source of assistance is the set of *HOWTO* documents available from the Linux Documentation Project (www.linuxdoc.org). Discussion forums and mailing lists are a good place to ask specific questions. Also, a search of web pages and discussion forums using a search engine such as Google (www.google.com) will, more often than not, reveal that someone else has already encountered the same problem and found a solution.

Conclusion

Linux is a good option for researchers on a budget because of its price, stability, flexibility, and power. With just a little effort and investigation, one can find a Linux solution for many common computer lab challenges. For researchers who do not have a large sum of money to spend on software and equipment, but do have the ability to do a bit of research, problem-solving, and experimentation, Linux provides an excellent tool for making the most of available resources when setting up a research lab.

References

Finney, S. A. (2001a). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods, Instruments, & Computers, 33*(1), 65-72.

Finney, S. A. (2001b). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, & Computers, 33*(2), 167-173.

MacInnes, W. J. & Taylor, T. L. (2001). Millisecond timing on PCs and Macs. *Behavior Research Methods, Instruments, & Computers, 33*(2), 174-178.

Ratcliff, R., Pino, C., & Burns, W. T. (1986). An inexpensive real-time microcomputer-based cognitive laboratory system. *Behavior Research Methods*, *Instruments*, *& Computers*, *18*, 214-221.

Table 1

Disk Geometry Reported by **parted** Program

```
(parted) print
Disk geometry for /dev/hda: 0.000-6532.866 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.031   3255.358  primary   FAT         boot
2       3255.359   6526.406  extended
5       3255.390   3561.284  logical   linux-swap
6       3561.315   5867.490  logical   ext2
(parted)
```

Disk geometry reported by the Linux partition editor program, parted, after the installation of RedHat Linux 7.1.

Table 2

Partition Listing from Linux **fdisk**

```
Disk /dev/hda: 255 heads, 63 sectors, 832 cylinders

Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System

/dev/hda1            33       415   3076416   80  Old Minix

/dev/hda2           416       832   3349552+   5  Extended

/dev/hda5           416       454    313236   82  Linux swap

/dev/hda6           455       748   2361523+  83  Linux
```

Partition information reported by **fdisk** after changing the partition type of the Windows partition (/dev/ha1) prior to installing DOS.

Table 3

Sample lilo.conf File for Booting DOS, Windows, and Linux

```
#This lilo.conf file is set up to boot:
# DOS from the first primary partition of the first hard drive (/dev/hda3),
# Windows from the second partition of the first hard drive (/dev/hda1), or
# Linux from the first extended partition of the first hard drive(/dev/hda5).
#
# Lines beginning with a hash mark (#) are comments that are
# ignored by lilo.
#
# You should consult the lilo man pages and HOWTOs in addition to this
# example to properly configure lilo for your machine.  Also see
# the grub man pages and HOWTOs for a different bootloader that can
# perform the same functions.
#
# Back up your existing /etc/lilo.conf file, then copy this file
# to /etc/lilo.conf and (logged in as root) issue the command lilo
# If error messages indicate that lilo did not install Linux into
# the boot loader successfully, edit the Linux section of
# /etc/lilo.conf to match the entry in your original lilo.conf file,
# then issue the command lilo again.  Otherwise your machine may not
# boot the next time you restart it.
#
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=1000
```

```
message=/boot/message

linear

default=Linux

#

image=/boot/vmlinuz-2.4.2-2

        label=Linux

        read-only

        root=/dev/hda6

#

# The next section boots Windows on the partition /dev/hda1

# Note that this partition is physically the second on the hard drive

# because it was moved using parted, but it still has the label

# hda1 because it was the first to be created.  The DOS partition

# /dev/hda3 is physically the first partition on the hard drive.

other=/dev/hda1

      optional

      label=Windows

        change                  #begin modifying the partitions

            #The following section makes the Windows partition visible,

            #in case its status was “hidden” from a previous DOS boot.

            partition=/dev/hda1

                  activate

                  set = DOS16_big_normal

            #The next section hides the DOS partition from Windows

            #so that Windows will think it is the first partition

            #when it is booted.

            partition = /dev/hda3

                  deactivate

                  set = DOS16_big_hidden
```

```
#
#The next section boots DOS on the partition /dev/hda3 (the partition that
#is physically at the beginning of the hard drive)
other=/dev/hda3
        optional
        label=DOS
        table=/dev/hda
        change                   #begin modifying the partitions
                                 #to hide the one not being booted
            #unhide and activate the DOS partition
            partition=/dev/hda3
                  activate
                  set = DOS16_big_normal
            #hide the Windows partition from DOS
            partition = /dev/hda1
                  deactivate
                  set = DOS16_big_hidden
```

Footnotes

[1] I have chosen to stay with the older LILO as my bootloader because of a single feature that GRUB has not yet implemented: the ability to set a certain operating system as the default to be loaded at the next reboot only, without resetting the global default operating system for subsequent reboots.

[2] If Partition Magic is used to do the initial repartitioning before installing Linux, this step could have been done at that point instead.

[3] For a nice set of simple command-line statistical utilities written in C, see |stat by Gary Perlman (http://www.acm.org/~perlman/stat/).

[4] Other alternatives for sharing files over a network include using Samba under Linux to create a shared drive that workstations running DOS or Windows can access, or using NFS to create a shared drive that can be accessed by workstations running Linux.