Linux has been a boon to computing in the developing world. In Pakistan we have used Linux productively in both academia and industry. We describe our experiences setting up a Linux-based teaching and research laboratory and present a survey of its use in industrial networking.

Linux and the Developing World

Shahid H. Bokhari and Rafeequr Rehman
University of Engineering and Technology, Lahore, Pakistan

FROM THE TRENCHES: Wolfgang B. Strigel, editor • wstrigel@spc.ca

The impact of Linux on the developing world in many ways exceeds its impact on industrialized nations. We have learned much about operating systems, networking, software tools, and languages from Linux. We have also had the opportunity to disseminate this information to industry in Pakistan. In particular, our work on networking, security, and firewalls[1] has assisted many Internet service providers here.

Linux provides the developing world access to the Unix/X Windows environment, the lingua franca of scientific and engineering computing, at essentially zero software cost. What is more, Linux runs on commodity IBM PC-compatible hardware, which is available at a fraction of the cost of comparable brand-name workstations. Many useful and important Linux applications such as compilers, networking software, and firewalls are available for free. The aspect of Linux that discourages its industrial use in affluent countries—the modest time investment required to set up and configure it—is of little concern in the developing world with its low labor costs. Linux has already had a major impact on the developing world's computing infrastructure, and its use and significance will likely grow in the next few years.

0740-7459/99/$10.00 © 1999

Research and Development

Researchers in the developing world find Linux useful for several crucial reasons. Prior to 1992, a researcher returning home after completing graduate or postdoctoral work in North America or Europe would effectively be moving from a world of Unix/X Windows to one dominated by MS-DOS/Windows. This was because the only commonly available Unix/X Windows systems ran on expensive workstations that institutions in developing countries could not afford.

This caused researchers much frustration and grief because they could not use familiar editors like vi or emacs. Standard tools like the Tex/Latex document preparation systems, the xfig drawing package, and the gnuplot plotting program were difficult to install and use on MS-DOS/Windows. Programs developed in C or Fortran, using the large address spaces of Unix compilers, were impossible to run on MS-DOS/Windows-based systems. Data sets, documentation, and manuscripts saved in Unix tar files required convoluted efforts to extract.

This all changed once a stable distribution of Linux became available. A full-fledged version of Unix—along with all of its standard tools, compilers, and utilities—was available for free and could be installed on cheap, commonly available IBM-PC compatibles. X Windows could be installed and the machines set up to support a collaborative working environment almost exactly like those available in research laboratories and graduate schools in the developed world.

In addition to providing continuity for researchers, Linux also exposed large numbers of students to the Unix/X Windows paradigm. This proved valuable as these students went on to work in the fast-developing fields of networking and Internet computing.

A Linux Laboratory

We have used Linux extensively for teaching and research in the Department of Electrical Engineering at the University of Engineering and Technology, Lahore. Our involvement with Linux began in 1992 when we installed the SLS distribution on a few 386-class PCs. We later migrated to the more stable Slackware distribution, which we still use today. Now we have nearly 100 machines running Linux—these range from diskless 386SX machines with 2 to 4 Mbytes of memory to 200-MHz Pentiums with 64 Mbytes of RAM and a 3-Gbyte hard drive. We described our early experiences in a 1995 article,[2] which still serves as a useful introduction to Linux.

Our current Linux project involves setting up a network of computers for undergraduate instruction in Unix and computer fundamentals (see the boxed text, "Building a Linux Laboratory", on p. 60). In this system, we have integrated 60 386SX machines with only floppy drives into a Linux system based on Pentium-133 servers. The 386SX machines, which work as character terminals, are connected to the servers via Ethernet. Students use floppy disks to load telnet programs into the 386SXs. IP (Internet protocol) numbers are assigned using a separate RARP (reverse address resolution protocol) server and are not hard-coded on the floppies; this eliminates duplication of IP numbers when students copy floppies from each other. Groups of 16 386SXs connect to hubs via unshielded twisted-pair wiring, and the hubs in turn connect to Pentium servers via thin-wire coaxial cable.

The many tools and compilers available on Linux make it ideal for teaching.

We have four servers each with 64 Mbytes memory and a 1.3-Gbyte hard disk. One is used as a domain name server, two serve as the actual computational servers to which students log on to execute their programs, and the fourth serves as backup. We have found that a single 133-MHz Pentium running Linux can comfortably support 60 students in an introductory programming course. In past years we used this laboratory to teach C and C++. We currently use it to teach freshmen basic programming skills using the Java language.

We are now extending our network to incorporate 12 AMD-586 166s and six Pentium 200s. These machines each have a minimum of 16 Mbytes of RAM and have their own hard disks. They can network-mount file systems from our existing servers and also use the network information system, which gives students the flexibility of working at any location. These new machines can run X Windows and thus can be used to run applets and Web browsers. Since they execute code locally, these machines put very little load on the servers and the network, and we do not envisage upgrading the servers in the near future.

Building a Linux Laboratory

A sophisticated laboratory can be built up with inexpensive PCs running Linux. A few large machines (486s, Pentiums, or Pentium IIs) can work as servers, storing user files and account information. These can be connected through "thinwire" (10Base2) Ethernet to other machines on which students log in and work. To reliably network more than a dozen or so machines, use a hub and 10BaseT (Unshielded Twisted Pair) wiring. A mixture of thinwire and UTP is also possible, as shown in Figure A.

Old 386 or 486 machines with little memory and without hard disks can serve for many years as character mode terminals. Newer machines can serve as powerful X workstations at a fraction of the cost of commercial products.

Various functions are typically provided by separate computers running Linux:

- A *compute server* enables users at terminals to log in to carry out their work.
- A *network file system* holds the user files that are available to all the machines on the network.
- A *domain name system* holds addresses for all the computers on large networks.
- A *network information system* maintains a central database of all user accounts, passwords, and so on.

One computer can provide all functions for a small network; large networks might employ several computers for each function. Network components include the hub, 10BaseT wiring, 10Base2 Ethernet, 386 or 486 machines with 2 Mbytes RAM and an Ethernet card, and PCs with 8 Mbytes or more of RAM and a hard disk to serve as the standalone X workstations.

The 386 and 486 machines can run Linux and be used as character mode terminals. The minimal Linux system requirements fit on a single floppy disk. Alternatively, a diskless system can use a boot PROM to load Linux from a server. Machines running DOS/Windows can also be used as terminals. These machines execute everything locally, referring to the servers only for user files, network addresses, and user account information. Diskless workstations can also be set up, although these tend to overload the network because all files must be fetched from a server.

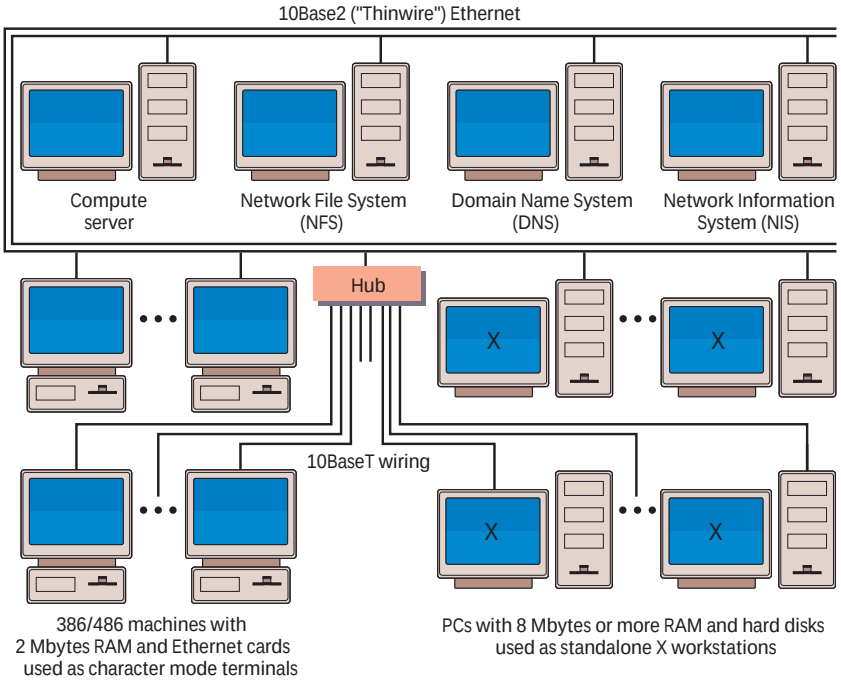


Figure A. A computing lab built with inexpensive PCs running Linux.

Teaching with Linux

The large variety of tools and compilers available on Linux makes it ideal for teaching courses in computer science and engineering. Excellent free compilers are available for C, C++, Objective C, Java, Pascal, Fortran, Modula-2 and -3, Ada, and Eiffel. Powerful tools for electronic circuit analysis and design are also available.

We are teaching an introductory course on Java programming using JDK 1.1.1. Many Linux-based editors are available; we chose Pico, one of the simplest and easiest for students to learn given that most have no prior computing experience. For more

sophisticated students we would likely use vi or emacs.

In prior years we taught C and C++ programming in this laboratory with very encouraging results. A course on object-oriented programming can be put together using the Gnu C compiler (which integrates C, C++, and Objective C), Java, and Smalltalk. The only cost involved is to configure the various compilers and write up instructions on their use. This encourages instructors to experiment with different languages and tools, which would be prohibitively expensive on a commercial operating system.

Of particular interest to many of our students is Rhide,[3] an excellent integrated program development environment that incorporates many useful features of Borland's Turbo C++ system without the annoying limitations of MS-DOS. Several large and complex programs have been developed in C++ in our department using Rhide.

One of the most interesting and useful applications we have encountered is the XMX X protocol multiplexer, developed by Brown University's John Bazik. This system allowed us to hold lectures on a network with the instructor's X Windows session echoed faithfully on each student's screen. We have described this in more detail elsewhere.[2]

We also set up an electronic lecturing system, even on character mode terminals, using a clever technique developed by Steve Moritsugu.[4] This allows everything that is typed on the instructor's terminal to be echoed on all student terminals. This proves very useful in the first few laboratory sessions of a course as students learn how to type in commands, change passwords, and so forth. It helps also to have a public address system in the lab, particularly with a large number of students.

Linux and the Internet

While setting up our laboratory, we also carried out an investigation on network security in Linux.[1] This was around the time that Internet usage was becoming popular in Pakistan, and experience gained in the laboratory proved beneficial to industry. Among the issues we tackled was how to set up Linux-based firewalls.

When an organization wishes to connect its internal computer network(s) to the Internet, it must resolve several security issues. Foremost is blocking access to the internal network (which may contain proprietary information) from malicious or even curious outsiders. A firewall is a computer, or a small network of computers, that sits between the Internet and a private network and implements a security policy.[5] Firewalls can operate in two ways:

- Packet filtering firewalls control the packets traveling through them based on the source/destination addresses and the ports and services that these packets use. Packet filtering is built into the Linux kernel; although useful, it does not provide control on a per-user basis.

Linux and Software Piracy

An interesting paradox emerges when we examine the relationship between Linux and software piracy. Linux is free and openly distributed, and provides an incredibly rich variety of tools to the cash-starved developing world. But this does not necessarily mean users will migrate in large numbers away from DOS/Windows to Linux.

The applications and programming environments available for free on Linux cannot match the glamour and glitz of commercial software based on Windows 95/98. Because of widespread software piracy in the developing world, legally free Linux has an ironic competition from illegally "free" pirated software. To some extent this works to the advantage of companies like Microsoft in that pirated software helps popularize Windows 95/98 and applications based on them in the developing nations, which constitute about half of the world's population.

However, this cannot continue indefinitely. We expect that in the near future, producers of popular commercial software packages will start incorporating strong encryption technology into their products. One possibility is to require a secure Internet connection to a metering facility that will extract immediate electronic payment when any software product is used. Shortly after the institution of such controls, users of pirated software will find themselves using outdated and non-upgradable versions of essential software tools.

Since a very large proportion of the users of pirated software in the developing nations are incapable of paying the "true" market cost of such products, such a move will have catastrophic consequences for users but will result in little additional revenue for the owners of software. In any case, instituting such controls will lead to explosive growth in the development of free or low-cost application tools for Linux that can, to some extent, match the functionality of commercial software.

Thus the paradox emerges: producers of commercial software and operating systems can suppress the adoption of Linux (at least in the developing world) by not attempting any serious controls on piracy. The moment electronic metering or other tamper-proof controls are instituted, the adoption of Linux will undergo a quantum leap. In anticipation of such developments, software houses and academic institutions in the developing world must maintain some level of investment in the study and use of Linux even if, unlike us, they are not presently interested in using it for productive work.

LINUX AND SUPERCOMPUTING

It is fashionable these days to talk about how Linux permits the creation of powerful supercomputers based on freely available low-cost hardware. Many such systems have been created in both developed and developing nations; some are described in this issue. We must remember, however, that these systems—which mainly use the PVM and MPI programming platforms—cannot be used for all types of parallel problems.

Furthermore, the problems to which they have been successfully applied require a fairly heavy investment of programmer time to obtain adequate performance. For example, nonuniform or unstructured problems, whose performance is extremely sensitive to the partitioning and mapping of the computational domain, require inordinate effort to execute efficiently on such machines. The interconnection technology of such systems (Ethernet or fast Ethernet) does not permit indefinite scalability, and the performance of such clusters saturates quickly for many nontrivial parallel applications.

We feel that exclusive reliance on such clustered supercomputing will be detrimental to the development of supercomputing in the long run. No interesting new supercomputer architectures are presently under development aside from the much-delayed Tera project (www.tera.com). Excessive reliance on "clusters" or "piles of PCs" will further distract the supercomputing community from squarely facing the serious, unsolved aspects of parallel computing.

Nonetheless, the main disadvantage of Linux clusters or piles—the labor-intensive nature of their programming—is a boon for the developing world, where low manpower costs make software development for low-cost parallel platforms a very attractive proposition.

♦ Proxy firewalls use software that acts as intermediary between an outside and an inside computer to manage precisely who and what is controlled and how.

A computer on the inside, for example, may wish to set up an http connection to a computer on the outside. An http proxy in the firewall will intercept this request and pass it to the outside computer if the setup configuration permits this. We have found that the free Firewall Toolkit (FWTK) for Linux developed by Trusted Information Systems (www.tis.com) is robust. FWTK provides different proxies for different applications with a central management and configuration file. All proxy servers are configurable separately and on a per-user basis. This gives very good control since we can allow or deny services based on user names and passwords.

We tested the FWTK firewall setup in our lab by splitting our local area network into two LANs and then connecting these through a firewall computer. Having established the utility of this approach, we then installed Linux-based firewalls at several industrial sites, where they have operated safely for several years. To evaluate performance, we tested the FWTK firewall for data throughput across LANs and found that it introduces no significant load to connect to the Internet.[1]

One extremely important facility that packet-filtering firewalls provide is IP masquerading: although hundreds of computers may reside in an organization's LAN or WAN, the connection to the outside world goes through a single IP number. The firewall must make all requests to the outside world appear as if they are emanating from a single IP number and must also redirect incoming packets to the appropriate internal computers. This permits an organization to access the Internet through one "true" IP number, while using any number of private IP numbers for its internal machines. Given the high cost of purchasing or renting IP numbers and the financial constraints in the developing world, Linux has had a major economic impact in this area alone.

Our implementations of IP masquerading were the first in our country and helped popularize this concept in many other academic locations as well as in industry. In fact, many Internet Service Providers in Pakistan now use IP masquerading to increase the number of dial-up connections for their end users once they run out of properly allocated IP addresses. Some initial problems arose in transparently passing some protocols, such as audio transmissions, but new patches and utilities supporting IP masquerading have mostly resolved these problems. Apart from ISPs, public and private educational institutions also find this technique useful and cost effective.

Since the cost of bandwidth to connect to the Internet is very high in the developing world as compared to the West, it is essential to optimize its use. Caching servers prove extremely valuable in this regard. An ISP typically uses a caching server to hold frequently requested Web pages for its users. For example, the caching server can save a popular Web page and supply it readily to all users who need it without having to repeatedly download the original through the Internet.

We found Squid (squid.nlanr.net) to be a very good caching proxy server for both small and large organizations using Linux to access the Internet. Linux running Squid on a Pentium 200 MHz with 64 Mbytes of RAM out-performs a similar machine running Windows NT and commercial proxy servers.

The newly introduced Transparent Proxy feature of the Linux kernel permits setting up a server so that all accesses are routed through a proxy. This is independent of any proxy settings that the user may have made in his browser. Using this feature in conjunction with Squid makes Linux the ideal caching server for the developing world.

We have also found another application, Radius (remote access dial-in user service), very useful.[6] Radius permits an ISP to keep its users' account, password, and billing information in one centralized location. This central database can then be used to control several remote-access servers, specialized hardware to which incoming phone lines connect. For example, one ISP in Pakistan maintains all user account and billing information on one Radius server even though it has numerous remote-access servers scattered through the country.

Outlook and Recommendations

We have learned a great deal from our long involvement with Linux:

♦ Memory has a greater impact on system performance than raw processor clock speed. At the time of this writing, 1 Mbyte of RAM costs about $1 (US). If you have $48 to spare, it is better to spend these on an additional 48 Mbytes of memory than on a faster processor chip.

♦ Always use an uninterruptible power supply (UPS). Power supply in developing nations is unpredictable. A loss of power while Linux is running can at best be an annoyance since Linux will check the file system when rebooting the system—a time-consuming process, given the size of today's hard disks. In the worst case, the user could lose important data or even damage the hard disk. When budgeting for computers, always plan for UPSs. At the very least, all servers should be powered through UPSs.[2]

♦ Never buy the fastest processor available. The fastest processor on the market at any point in time is always overpriced in relation to the true computational power it provides. If you are setting up a laboratory with multiple computers, it is always cost-effective to buy the processor that preceded the current "star" processor. You can often save enough money to increase the total number of machines by 25 percent.

♦ You can manage without air conditioning. Most of the developing world has very hot summers, but air conditioning is a big expense in terms of both equipment and running costs. These costs can be prohibitive, especially for state colleges and universities, but can be minimized by

- restricting air conditioning to small areas where servers are kept,
- not using temperamental, high-MHz processors that are more heat-sensitive,
- using monochrome monitors, if possible (they generate less heat), and
- shutting down laboratories in the hottest parts of the year (June–Sept for us). Be warned, however, of the trend toward processors with ever-increasing MHz, and away from monochrome monitors. Laboratories with many machines may in the end need to be air-conditioned.

♦ Do not use 10Base2 ("thinwire") Ethernet for more than a dozen machines. 10Base2 Ethernet runs a single coaxial cable through all computers in the network and is a convenient, cost-effective way to create a small network. It is not recommended for a large network, especially in a university laboratory, because a single malfunctioning or disturbed connection can bring down the entire network. The alternative, 10BaseT Ethernet, requires that you invest in a hub (see the boxed text, "Building a Linux Laboratory," on p. 60). This expense proves worthwhile because 10BaseT wiring and connectors are far more reliable and the failure of a single connector affects only one machine.

Linux is the only way most developing nations have to legally access modern and sophisticated software tools.

Linux gives users in the developing world, who have and will continue to have severe financial constraints, access to an entire universe of software tools exactly the same as those used in advanced nations under Unix and X Windows. In this respect alone Linux has propelled the development of computer science and engineering in the poorer nations. Linux is the only way most developing nations have to legally access modern and sophisticated software tools, compilers, and programming environments. ❖

ACKNOWLEDGMENTS

No article on Linux can be complete without acknowledging the contributions of Linus Torvalds and the many thousands of Linux enthusiasts worldwide who have created a shared, cost-free artifact of such epic proportions.

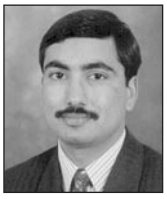
Our computer laboratory has been set up with donations by the Syed Maratib Ali Trust, the Babar Ali Foundation, and UET alumni worldwide. Many faculty, staff, and students have volunteered their efforts to help set up our network, and gained our everlasting gratitude. Rafeeq ur Rehman is supported by a doctoral fellowship from the Ministry of Science and Technology, Government of Pakistan. Additional support for this work was provided through grants from the Directorate of Research Extension and Advisory Services of the UET. We are grateful to Altaf Khan for his valuable advice during the preparation of this manuscript.

REFERENCES

1. R. ur Rehman, "Security and Firewalls in Linux Networks," master's thesis, Dept. of Electrical Eng., Univ. of Eng. and Technology, Lahore, Pakistan, June 1997.
2. S.H. Bokhari, "The Linux Operating System," *Computer*, Vol. 28, No. 8, Aug. 1995, pp. 74-79.
3. R. Höhne, "The Rhide Integrated Program Development Environment," www.tu-chemnitz.de/~sho/rho/rhide/rhide.html.
4. S. Moritsugu, "Linux at Rancho Santiago College," *Linux J.*, No. 45, Jan. 1998, pp. 48-50.
5. W.R. Cheswick and S.M. Bellovin, *Firewalls and Internet Security*, Addison Wesley Longman, Reading, Mass., 1994.
6. R. ur Rehman, "Distributed Network Security through RADIUS," *System Administrators' J.*, Vol. 6, No. 10, Oct. 1997, pp. 63-70.

About the Authors

Shahid H. Bokhari received a BSc in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, and an MS and PhD in electrical and computer engineering from the University of Massachusetts, Amherst. He is a professor of electrical engineering at the University of Engineering and Technology, Lahore, and is associated with the Institute for Computer Applications in Science and Engineering (ICASE) in Hampton, Virginia. His research interests lie in parallel and distributed computing. Bokhari is a member of ACM and the IEEE Computer Society and is an IEEE Fellow.



Rafeeq ur Rehman received BSc and MSc degrees in electrical engineering from the University of Engineering and Technology, Lahore. He is currently pursuing a PhD from the same institution, working on distributed computing and computer networks. He has helped many organizations in Pakistan set up LANs and WANs, drawing on his master's research on Linux security.

Readers may contact the authors at the Dept. of Electrical Engineering, Univ. of Engineering and Technology, Lahore 54890, Pakistan; sbokari@computer.org and rehman@pol.com.pk.