The journal of Apple technology.

Volume Number: 13 (1997)
Issue Number: 10
Column Tag: Tools Of The Trade

Why MkLinux?

by Rich Morin

Or, is grep really worth 500 MB of disk space?

So What *Is* MkLinux?

MkLinux is a port of the GNU System to the Power Macintosh. It provides a Unix-like user and programming environment, with hundreds of useful commands (software development and document preparation tools, network servers and tools, and other utilities. With very few exceptions (e.g., the boot code), all of the source code for MkLinux is freely redistributable.

The MkLinux port is based on the Linux 2.x kernel, which is used as a "single server" for the Mach 3 microkernel. Thus, the Linux kernel runs as a user-mode process, handling OS "personality" issues, while the Mach microkernel handles low-level I/O and resource allocation tasks. This two-part design has assorted benefits, including the ability to debug a new Linux server (as an application) while running the system on a older (trusted) server.

Because MkLinux has pre-emptive multitasking and runs as a "native" OS, it makes quite efficient use of the Power Macintosh hardware. A Power Mac 6100, for instance, is capable of serving as the local "unix box" for a dozen or more casual users and programmers. Aside from providing access to the normal range of unix tools, MkLinux supports a wide variety of network services, including FTP, HTTP, NFS, Telnet, and the X Window System.

Once a MkLinux system has booted, it is "just" a Linux system. That is, it has all of the standard Linux user and programming interfaces, system administration tools, etc. The big difference, however, is that MkLinux is running on an integrated hardware platform. Consequently, MkLinux system installation does not lose the administrator in a twisty maze of DMA vector and IRQ settings, jumpers, and other Intel-specific annoyances.

History and Current Status

Although MkLinux has been under development for a few years, it has only been available to the general public for a short while. Apple's first public announcement concerning MkLinux was made at the Free Software Foundation's First Conference on Freely Redistributable Software (February 1996). Since that time, Apple has issued several MkLinux Developer Releases. DR3, for instance, is expected to be available by the time you read this article.

MkLinux has never been a large-scale project for Apple Computer. Staffing has never exceeded five engineers, split between Apple and The Open Group Research Institute. The MkLinux Reference Release, although sponsored and assisted by Apple, was produced by Prime Time Freeware. Through the efforts of volunteers, large numbers of applications and drivers appear on MkLinux-related FTP sites. If Linux is the system where "everything is done by someone else", MkLinux is certainly well in line with this tradition.

The MkLinux user community is served by email lists, web sites, and other Internet-based facilities. The email lists

are quite active; more than 5000 subscribers are currently registered. They handle MkLinux announcements, setup and programming issues, etc. The high signal-to-noise ratio of the lists reflects the cooperative attitude of the MkLinux community. The Apple MkLinux web site http://www.mklinux.apple.com contains announcements, news, online help, and pointers to other Internet-based resources.

Who Uses MkLinux?

The Macintosh is very popular in academia, whether in computer labs, professor's offices, or dormitory rooms. By making Linux available for the Power Macintosh, Apple allows these academic institutions (and associated individuals) to make better use of existing hardware. For instance, a computer science graduate student might write a device driver for MkLinux, then switch back to Mac OS to create the diagrams and write up a paper describing it.

As hinted above, MkLinux is also finding use as a network server. NuBus-based Power Macs are not considered very powerful by today's standards, but they perform quite well as MkLinux servers. In fact, a Power Mac 6100/60 is roughly equivalent in performance to an early Sun SparcStation 5 or a 90 MHz Pentium system. If the 6100 is already on hand (or cheaply available), it can be an economical and very convenient alternative to an Intel or Sparc-based system.

We are also starting to see quite a bit of interest from the Apple developer community. Developers have been quick to realize that MkLinux provides a very economical way to look at a Mach/unix system on the Power Macintosh. Better yet, both the source code and a large body of reference materials are available.

Apple's original motivation in sponsoring MkLinux, by the way, came from its Higher Education sales force. It seems that they got tired of hearing about the fact that Intel-based PCs were more versatile (running both Windows and Linux). So, when the MkLinux skunk works project got started, the Higher Ed folks were quick to support it.

MkLinux Basics

If you are already familiar with Intel-based Linux systems, feel free to skip down to the next section: a running MkLinux system is essentially indistinguishable from a running Intel-based Linux system. If you have used other unix-based systems, all you need to know is that (Mk)Linux provides extended versions of most traditional BSD and System V commands.

If you're still with us, you're probably wondering how much of a learning curve you're facing before you can be a productive MkLinux user and programmer. As usual, the news is mixed: the bad news is that it could take you years to learn everything about the entire system. The good news is that you can master the basics in a few hours and that you don't really need to know everything about MkLinux to be quite productive on it. So, let's get started...

Like MS-DOS, MkLinux has a command-oriented user environment. Although X Windows provides the ability to run graphical applications, it does not have the pervasive GUI-based semantics, let alone the consistent user interface guidelines, found in Mac OS. Hence, MkLinux windows tend to be "glass teletypes", used to communicate with a command line interpreter.

Unlike MS-DOS, however, MkLinux supples several command line interpreters, or "shells", along with hundreds of shell-oriented commands and "little languages". By combining these tools, either on the command line or as saved "shell scripts", you can make MkLinux do some pretty amazing things.

Let's say that one of your directories contains several hundred files, each of which contains one or more C routines. You want to look at some examples of how the routine "foo" is used, so you ask the MkLinux "grep" command to take a look:

```
rdm@mklinux 1: grep foo *.c
```

```
bar.c:  x = foo(...);
baz.c:  y = foodp(...);
baz.c:  z = foo(...);
...
```

Although the first line looks just fine, the second line shows that we weren't sufficiently precise in our search. So, we use the "-w" flag to tell grep to look for the "word" foo:

```
rdm@mklinux 2: grep -w foo *.c
bar.c:  x = foo(...);
baz.c:  z = foo(...);
...
```

There seem to be quite a lot of matching lines, however; let's just get a list of the file names:

```
rdm@mklinux 3: grep -lw foo *.c
bar.c
baz.c
...
```

How many files are there? Let's ask the MkLinux word count (wc) utility for a count:

```
rdm@mklinux 3: grep -lw foo *.c | wc -l
42
```

Now let's suppose that we want to copy all of the matching files off to a separate directory, for detailed examination and/or modification. MkLinux lets us do this very easily:

```
rdm@mklinux 4: mkdir hax
rdm@mklinux 5: cp `grep -lw foo *.c` hax
```

There are hundreds of commands in the MkLinux repertoire, yielding a staggering number of combinations. And, because sequences of commands can be saved as "shell scripts", creating new commands is a trivial process. MkLinux shells have most of the common programming constructs, including variables, control flow operators, etc. If you want more programming power, however, you can always drop into a language such as awk or perl.

The creation and (re-)use of simple commands is fundamental to the MkLinux way of doing things. Rather than try to create monolithic programs that can do every conceivable task, MkLinux programmers create small, general-purpose tools and combine them into task-specific commands and scripts.

In the example above, for instance, cp has no need to know anything about string searching and grep has no need to know anything about file copying. The combination, however, is able to do a rather powerful and specialized function (string-based file copying) with little effort.

Legal Issues

As an Apple developer, you are probably familiar with GCC, GDB, GNU Emacs, and other tools that have emerged from the GNU Project. As an implementation of the GNU System, MkLinux contains hundreds of these tools. In addition, it includes a great deal of BSD (Berkeley Software Distribution), Mach, and X Window System code. In short, MkLinux is a compilation of compilations, including a variety of software and associated legal restrictions.

Software developed by the GNU Project is (almost always) covered by the terms of the GNU General Public Licence (GPL). In addition, many programs developed by third parties (e.g., GhostScript and Perl) are also available

under the GPL. Thus, if you plan to use MkLinux (or any other GNU software), a basic understanding of the GPL is advised.

The GNU GPL allows both non-commercial and commercial distribution, in either source or binary format, of materials it covers. It requires, however, that distributors of binary code make the source code economically available, upon request. It also prohibits any other restrictions (including collection copyrights) from being placed on the materials.

Thus, although you are free to examine, modify, redistribute, and use GPLed code, you are not free to turn it into proprietary code. In general, software developed using GNU tools is not affected by the GPL, save that use of libraries may "contaminate" developed code. For detailed information on the GNU Project and the GNU GPL, see the GNU web site http://www.gnu.org.

Much of the remaining freeware in the MkLinux distribution is covered by "University-style" licenses, such as those used by CMU, MIT, and UC Berkeley. These licenses require that the source code retain a notice showing its origins, but say nothing restricting the code's inclusion into commercial products.

Finally, you should expect to find a few "unique" licenses, ranging widely in both style and terms. You are on your own when dealing with these notices. Although these licenses normally allow free use within a given organization, their terms for distribution may prohibit external distribution, even for non-commercial purposes. So, be sure to read the fine print!

Porting Issues

Because MkLinux is closely based on Linux, it is able to build most Linux source archives with little trouble. The problems that do surface tend to stem from basic architectural incompatibilities such as byte ordering or the need for particular hardware. Finally, because of the Mach-based internal structure of MkLinux, Linux driver code does not port at all trivially.

Most Linux programs are written in an architecture-independent manner. At worst, they contain a few #ifdef statements. A few Linux programs, however, make assumptions about the ordering of bytes within words, etc. Because PowerPC CPUs address differently than Intel CPUs, this can cause programs to malfunction. There is also an assumption, on Intel-based platforms, that variables of type char are signed. MkLinux assumes that chars are unsigned, but a compiler option can be used to change this assumption.

If the source code comes from another variant of unix (e.g., BSD or System V), things can get a bit trickier. Although most unix system calls and library functions are standardized, each flavor of unix tends to have a few unique (or modified) interfaces. Linux is quite popular, however, so you should expect to find Linux-oriented #ifdef statements in most common freeware packages.

Porting non-trivial software from Mac OS, on the other hand, is likely to be a real challenge. The MkLinux system calls and library functions have many of the same capabilities of Mac OS routines, but their exact nature, names, and calling sequences are apt to to be very different. Be prepared to look up a lot of function definitions!

Worse, the event-driven model favored by Mac OS applications is not found in most MkLinux programs. Some window-based applications may be written in an event-driven manner, but that's about it. And, although X may look a bit like the Finder, it is a radically different environment, sharing few of the Finder's basic assumptions and rules. In short, you should treat MkLinux as a Linux system that happens to run on a Power Macintosh, rather than a weird variant of Mac OS.

MkLinux Resources

As noted above, MkLinux is well served by email lists, FTP sites, and web servers. The best starting point is Apple's MkLinux web server http://www.mklinux.apple.com. Look around this site for general information, late-breaking news, mailing lists, online help, and links to related sites. As the MkLinux community grows, volunteers will take up more and more of the burden of disseminating MkLinux-related information, but the Apple web site is likely to remain as the best starting point for any search.

The Linux community is well served by printed documentation, software collections, and online information resources. With the exception of some kernel- and hardware-related material, most of these resources are quite applicable to MkLinux. More generally, much of the unix literature is relevant to MkLinux. After all, MkLinux is basically a free version of unix.

All modesty aside, however, the definitive work on MkLinux is "MkLinux: Microkernel Linux for the Power Macintosh" (Prime Time Freeware, 1997, ISBN 1-881957-24-1, $50 MSRP). This Apple-sponsored product combines a 360-page introductory and reference manual with a MkLinux distribution CD-ROM and an HTML-basedreference CD-ROM, covering Linux, Mach, NeXT, and the Power Macintosh. For more information, visit the Prime Time Freeware web site http://www.ptf.com.

Rich Morin, rdm@ptf.com, is the editor of "MkLinux: Microkernel Linux for the Power Macintosh", published by Prime Time Freeware (http://www.ptf.com). A 30-year veteran of the computer industry, Rich writes for both SunExpert and UNIX Review magazines. His home system is a three-headed Power Macintosh 7100/80, running Mac OS. It is, however, networked to several local UNIX (MkLinux and SunOS) boxes.