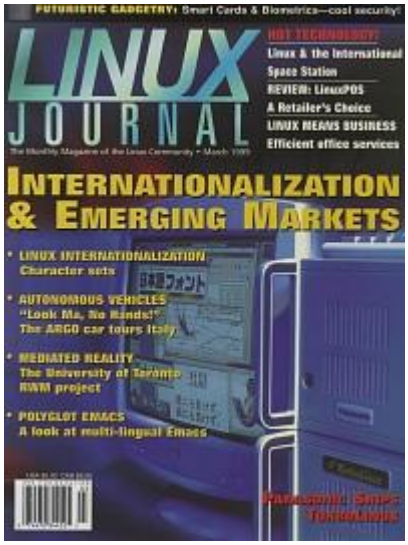
Linux Journal Issue #59/March 1999



Focus

Features

A look at multilingual Emacs.

Forum

Smart Cards and Biometrics *by David Corcoran, David Sims and Bob Hillhouse*

The cool way to make secure transactions.

Linux for the International Space Station Program *by Guillermo Ortega*

An overview of two applications for spacecraft and why these applications are being run on Linux.

LJ Talks to Chris Brown of Learning Tree International *by Marjorie Richardson*

Linux enters the mainstream as companies such as Learning Tree and Caldera offer training courses for Linux. Here Learning Tree tells us why they are doing it.

Reviews

LinuxPOS, An Opportunity Waiting to Happen *by Brian Walters*
Red Hat LINUX Secrets, Second Editon *by Duane Hellums*

Columns

Focus on Software *by David A. Bandel*
Linux in Education Linux in a Public High School *by Andrew Feinberg*

Another high school student brings Linux and the Internet to his fellow students.

The Cutting Edge The Linux Router Project *by David Cinege*

A look at one of the fastest growing Linux distributions, that you may never actually see.

Linux Means Business Cost Effective Services for the Office *by Kim Henderson*

How the Linux operating system made possible cost-effective company e-mail and created opportunities for adding useful services.

At the Forge Creating a Web-Based BBS, Part 3 *by Reuven M. Lerner*

Mr. Lerner shows us how to add a full-text search to our BBS.

Departments

Letters to the Editor

More Letters to the Editor

Best of Technical Support
Stop the Presses Partners—Pacific HiTech and Panasonic *by Marjorie Richardson*
New Products

Strictly On-line

The K Desktop Environment, Version 1 *by Bill Cunningham*
Linux Network Toolkit *by Russell J. T. Dyer*

Linux and the EURO Currency: Toward a Global Solution *by Guylhem Aznar*

Mr. Aznar talks about problems and solutions to adding the EURO symbol to the keyboard.

Archive Index

Advanced search

Internationalization and Emerging Markets

Marjorie Richardson

Issue #59, March 1999

Linux has long been used in countries where languages other than English are spoken and the needs of those users must be addressed.

As Linux becomes even more widespread around the world, we felt a focus on internationalization would be timely. Linux has long been used in countries where languages other than English are spoken and the needs of those users must be addressed. Not only is there the difficulty of translations, but also different character sets. In addition to outputting program messages in the local language, a way to edit and manipulate multilingual text is a must.

Many of these topics have been addressed and our feature articles tell us how. The latest version of Emacs includes multilingual extensions for support of text entry in different languages, including those based on ideographic characters. The GNU gettext system provides the tools for developers to output messages in multiple languages. There is even work to get the euro symbol onto the keyboard (see the “Strictly On-line” article “Linux and the EURO Currency: Toward a Global Solution” by Guylhem Aznar).

This month, we also focus on emerging markets—not those products that are announcing Linux support each day, but truly new products which have not been available for any operating system. Computer driven vehicles, wearable computers, smart cards and space station applications can all be found running under the Linux operating system. Join us as we enter the future with Linux.

Marjorie Richardson, Editor

Alphabet Soup: The Internationalization of Linux, Part 1

Many problems exist when more than one character set is needed. Standards are being developed to deal with these problems. In this first of a two-part

series, Mr. Turnbull takes a look at just what internationalization means to all areas of the Linux world.

by Stephen Turnbull

Autonomous Vehicles

The future is here—cars that can be driven by a computer while we nap. And what operating system drives the computer? Linux, of course. Read all about it in this article on the ARGO Project being conducted at the University.

by Massimo Bertozzi, Alberto Broggi and Alessandra Fascioli

Internationalizing Messages in Linux Programs

Program developers wishing to localize the messages output from their applications will want to read this article about the GNU gettext system. This system offers a set of tools and libraries which enable multilingual programming.

by Pancrazio de Mauro

Mediated Reality: University of Toronto RWM Project

Is it real or is it mediated? This month, Dr. Mann shows us how to change our reality to substitute pleasing pictures for unwanted advertising or just jazz up an otherwise humdrum scene—all by using his wearable computer that looks just like an ordinary pair of sunglasses.

by Dr. Steve Mann

Polyglot Emacs 20.4

Preparing text that includes multiple languages no longer needs to be a frustrating job. Learn how to do it easily with the new Emacs that includes multilingual extensions (Mule).

by Jon Babcock

Archive Index Issue Table of Contents

Advanced search

Alphabet Soup: The Internationalization of Linux, Part 1

Stephen Turnbull

Issue #59, March 1999

Mr. Turnbull takes a look at the problems faced when different character sets and the need for standardization.

What is Linux? Since you are reading this in the *Linux Journal*, you probably already know. Still, it is worth emphasizing that Linux is an open-source software implementation of UNIX. It is created by a process of distributed development, and a primary application is interaction via networks with other, independently implemented and administered systems. In this environment, conformance to public standards is crucial. Unfortunately, internationalization is a field of information processing in which current standards and available methods are hardly satisfactory. The temptation to forfeit conformance with (international) standards in favor of accurate and efficient implementation of local standards and customs is often high.

What is internationalization? It is not simply a matter of the number of countries where Linux is installed, although that is certainly indicative of Linux's flexibility. Until recently, although their native languages varied widely, the bulk of Linux users have been fluent in certain common not-so-natural languages, such as C, sh and Perl. Their primary purpose in using Linux has been as an inexpensive, flexible and reliable platform for software development and provision of network services. Of course, most also used Linux for text processing and document dissemination in their native languages, but this was a relatively minor purpose. Strong computer skills and hacker orientation made working around the various problems acceptable.

Today, many new users are coming to Linux seeking a reliable, flexible platform for activities such as desktop publishing and content provision on the World Wide Web. Even hackers get tired of working around software deficiencies, so now a strong demand exists for software to make text processing in languages other than English simple and reliable, and permitting text to be formatted according to each user's native language and customs.

This process of adapting a system to a new culture is called localization (abbreviated L10N). Obviously, this requires provision of character encodings, display fonts and input methods for the input and display of the user's native language, but it also involves more subtle adjustments to facilities such as the default time system (12 hour or 24 hour) and calendar (are numerical dates given MM/DD/YY as in the U.S., or YY/MM/DD as in the international standard, or DD/MM/YY?), currency representation and dictionary sorting order. APIs for automatic handling of these issues have been standardized by POSIX, but many other issues, such as line-wrapping and hyphenation conventions, remain. Thus, localization is more than just providing an appropriate script for display of the language and, in fact, more than just supporting a language. American and British people both use the same language as far as computers can tell, but their currency symbols are different.

Localization is facilitated by true internationalization, but can also be accomplished by patching or porting any system *ad hoc*. To see the difference, consider that a Chinese person who wishes to deal with Japanese in the Microsoft Windows environment has two choices: dual booting a Japanized Windows and a Sinified Windows, or using the rather unsatisfactory and generally unsupported by applications Unicode environment. This is a localization; it is non-trivial to port applications from Japanized Windows to Sinified Windows, as the same binaries cannot be used. In an internationalized setup, one would simply need to change fonts, input methods and translate the messages; these would be implemented as loadable modules (or separate processes). With respect to applications, the situation in Linux is, at best, somewhat better (especially from the standpoint of Asian users). However, the future looks very promising, because many groups are actively promoting internationalization and developing internationalized systems for the GNU/Linux environment.

Internationalization (abbreviated I18N) is the process of adapting a system's data structures and algorithms so that localizing the system to a new culture is a matter of translating a database and does not require patching the source. Of course, we would prefer the binaries to be equally flexible, but for reasons of efficiency or backward compatibility, localized versions may implement different data structures and algorithms. Although internationalization is more difficult than localization, once it is complete, the process of localizing the internationalized software to a new environment becomes routine. Furthermore, localization by its nature is not a strong candidate for standardization, because each new system to be localized to a particular environment brings its own new problems. Internationalization, on the other hand, is by definition a standard independent of the different cultural environments. An obvious extension is to jointly standardize those facilities common to many systems.

Internationalization can be contrasted with multilingualization. Multilingualization (abbreviated M17N) is the process of adapting a system to the simultaneous use of several languages. Obviously more difficult than localization or even internationalization, multilingualization requires that the system not only deal with different languages, but also maintain different contexts for specific parts of the current data set.

Note that the operating system can be localized, internationalized or multilingualized while some or all applications are not, and vice versa. In a certain sense, Linux is a multilingual operating system; the kernel presents few hindrances to use of different languages. However, most utilities and applications are limited to English by availability of fonts and input methods, as well as their own internal structures and message databases. Even the kernel panics in English. On the other hand, GNU Emacs 20, both the FSF version and the XEmacs variant, incorporate the Mule (MUlti-Lingual Extensions Emacs) facilities (see “Polyglot Emacs” in this issue). With the availability of fonts and, where necessary, internationalized terminal emulators, Emacs can simultaneously handle most of the world's languages. Many GNU utilities use the GNU **gettext** function (see “Internationalizing Messages in Linux Programs” in this issue), which supports a different catalog of program messages for each language.

The Linux Kernel

The foundation is the operating system kernel. The main condition imposed on the functionality of the kernel is that it must treat natural language text data as binary data. Data passed between processes should not be altered gratuitously in any way. For many kernel functions (e.g., scheduling and memory management), this is already a requirement to which most systems conform.

Historically, computer text communications used 7-bit bytes and the ASCII character set and encoding. Often, communications software would use the high bit as a “parity bit”, which facilitates error detection and correction. Some terminal drivers would use it as a “bucky-bit” or to indicate face changes, such as underlining or reverse video. Although none of these usages are bad per se, they should be restricted to well-defined and well-documented interfaces, with 8-bit-clean alternatives provided.

Unfortunately, the seven-significant-bits assumption leaked into a lot of software, in particular implementations of electronic mail, and now has been enshrined in Internet standards such as RFC-821 for the Simple Mail Transport Protocol (SMTP) and RFC-822 which describes the format of Internet message headers for text messages. Software that transparently passes all data and does not perform unrequested transformations is called “8-bit clean”. Note that endianness is now an issue in text processing, since character sets such as

Chinese, Japanese, Korean and the Unicode character set use at least two-byte integers to represent some or all characters. Hencefoward, any mention of 8-bit cleanness should be taken to imply correct handling of endianness issues as well.

One might also want kernel log and error messages to be localized. Although error messages are rather stylized and easy to learn, they are also easy to translate. The main objection to doing this is efficiency. Compiling all known languages into the kernel would make it huge; localizing the kernel would mean maintaining separate builds for each language. A more subtle problem is the temptation to avoid new features that would require translating more messages, or to do a hasty, inaccurate translation. Finally, separate message catalogs would require loading at some point, introducing more possibilities for bugs. Even worse, a system crash before loading the catalog would mean no error messages at all.

File Systems

The next layer is file systems which are normally considered part of the kernel. In cases like Linux's /proc file system, they are inseparable from the kernel. File systems are a special case because the objects they contain inherently need to be presented to users and input by them. Hence, it is not enough for the programs implementing file systems to be 8-bit clean, since the directory separator, /, has special meaning to the operating system. This means that trying to use Japanese file names encoded according to the JIS-X-0208 standard may not work because some of those two-byte characters contain the ASCII value 0x2F as a second byte. If the file system treats path names as simply a string of bytes, such characters will be corrupted when passed to the file system interface in function arguments.

One can imagine various solutions to the problem such as redesigning file system calls to be aware of various encodings, using a universal encoding such as Unicode, or removing dependence of the operating system on such special characters by defining a path data type as a list of strings. However, these solutions would preclude backward compatibility and compatibility with alien file systems and would be something of a burden to programmers. Fortunately, a fairly satisfactory solution is to use a transformed encoding that sets the eighth bit on non-ASCII characters. The major Asian languages have such encodings (EUC), as does Unicode (UTF-FSS, officially UTF-8). By definition the ISO-8859 family of encodings, all of which contain U.S. ASCII as a subset, satisfy this constraint without need for transformation. Using a transformation format is rarely a burden on users, as they generally need not be aware of which encoding format their language is using. However, this difficulty can still apply to mounting alien file systems, especially MS-DOS and VFAT file systems, where Microsoft has implemented idiosyncratic encodings such as Shift-JIS for their

localized extensions of DOS and MS Windows. A partial directory listing of a Japanized Windows 95 file system mounted as an MS-DOS file system on my Linux system is shown in Figure 1, a Dired buffer in Mule. Note the error messages at the top of the buffer. **ls** is unable to find a file name it just received from the file system. When the directory listing is made, one pass generates a list of file names, which includes Shift-JIS-encoded Japanese names. When the list is passed back to the file system to get the file properties, some of the octets of Japanese characters are interpreted as file system separators. Thus, the Japanese character is not found and the error message results. The VFAT file system does not exhibit this problem in this case; I am not sure why. Most characters are passed unscathed, as you can see.

Figure 1. Japanized Windows 95 File System on Linux

This principle applies generally to other system processes (init, network daemons, loggers and so on). As long as the programs implementing them are 8-bit clean, use of non-ASCII characters in comments and strings in configuration files and the like should be fairly transparent, as long as file-system-safe transformation formats of standard encodings are used. This is true because keywords and syntactically significant characters have historically been drawn from the U.S. ASCII character set and, in particular, U.S. English. This is unlikely to change because the historic dominance of the U.S. in computer systems manufacturing and distribution means that most programming and scripting languages are English-based and use the ASCII character set. An interesting exception is APL; because of its IBM heritage, it is based on EBCDIC, which contains many symbols not present in ASCII.

The major standards for programming languages, operating systems (POSIX), user interfaces (X) and inter-system communication (RFC 1123, MIME and ISO 2022) specify portable character sets which are subsets of ASCII. Protocol keywords are defined to be strings from the portable character set, and where a default initial encoding is specified, it is U.S. ASCII or its superset, ISO-8859-1. Note that in most cases a *character set* is specified, for example, in C and X. IBM mainframes must support a portable character set which is a subset of ASCII, but those characters will be encoded in EBCDIC.

The only exception is ISO 2022 which specifies neither a portable nor ASCII character set as default; however, even there the influence of ASCII is extremely strong. The 256 possible bytes are divided into “left” (eighth bit zero) and “right” (eighth bit one) halves of 128 code points each and within each half, the first 32 code points (0x00 to 0x1F) are reserved for control characters while the remaining 96 may be printable characters. Further, positions 0x20 and 0x7F have reserved interpretations as the space and delete characters respectively

and may not be used for graphic characters, while 0xA0 and 0xFF are often left unused.

User Interfaces

The next layer is user interfaces, such as the Linux console and X. Here, the strong preference is for a primitive form of multilingualization, allowing arbitrary fonts to be displayed and text to be input in arbitrary encodings via configurable mappings of the keyboard. Both the Linux console and X provide these features, although the Linux console does not directly support languages with characters that cannot be encoded in one byte. They need not have more sophisticated mechanisms, because users rarely deal directly with them; application developers will build user-friendly interfaces on top of these toolkits. On the other hand, they should be as general as possible, so that the localizations can be as flexible as possible.

Applications

The next layer is applications, including system utilities. Here, things become much more complicated. Not only is it desirable that they issue messages and accept input in the user's native language, but must also handle non-trivial text manipulations like sorting. Of course, their entire purpose may be text manipulation, e.g., the text editor Emacs or the text formatter TeX.

One example of this complexity is that even where languages have characters in common, the sorting order is typically different. For example, Spanish and English share most of the Roman alphabet and both can be encoded in the ISO-8859-1 encoding. However, in English the names Canada, China and Czech Republic sort in that order, but in Spanish they sort as Canada, Czech Republic and China, because Spanish treats “ch” as a single entity, sorting after “c” but before “d”. Although Chinese, Japanese, Korean and to some extent Vietnamese share the ideographic characters that originated in China, they have very different ideas about how those characters are sorted.

Inter-system Communication

The outermost layer, beside the user-to-system interface, is inter-system communication. This layer has all the problems already mentioned, plus one more. Within a single system, specifying how to handle each language can be done implicitly; when a language is recognized, the appropriate version of some subsystem handles it. However, when communicating with another system, a mechanism for specifying formats must be present. Here, the MIME (Multipurpose Internet Multimedia Extensions) formats are crucial. Where possible, a means to negotiate the appropriate format for the communication

should be provided as in HTTP, the hypertext transport protocol which is the foundation of the World Wide Web.

Components of Localization

Localizing an application means enabling it to display, receive input and modify text in the preferred language of the user. Since this is usually the user's native language, we will also write native language support (NLS) for localization.

Text Display

The most basic capability is text display. Merely discussing text display requires three concepts: character set, encoding and font. A language's *character set* is those characters used to form words, phrases and sentences. A character is a semantic unit of the language and the concept of “character” is quite abstract. Computers cannot deal directly with characters; they must be encoded as bit strings. These bit strings are usually 8 bits wide; strings of 8 bits are called *octets* in the standards. “Byte” is not used because it is a machine-oriented concept; octets may refer to objects transmitted over a serial line, and there is no need for the hosts at either end to have facilities for handling 8-bit bytes directly. Most Linux users are familiar with the hexadecimal numbering system and the ASCII table, so I will use a two-hex-digit representation of octets. For example, the “Latin capital letter A” will be encoded as 0x41.

Human readers do not normally have serial interfaces for electronic input of bit strings; instead, they prefer to read a visual representation. A *font* is an indexed set (not necessarily an array, because there may be gaps in sequences of the legal indices) of *glyphs* (character shapes or images) which can be displayed on a printed page or video monitor. The glyphs in a font need not be in a one-to-one correspondence with characters and do not necessarily have semantic meaning in the native language. For example, consider the word “fine”. As represented in memory, it will consist of the string of bytes “Ox66 Ox69 Ox6E Ox65”. Represented as a C array of characters, it would be “fine”, but as displayed after formatting by TeX in the PostScript Times-Roman font, it would consist of **three** glyphs, “fi”, “n” and “e” as shown in Figure 2.

Figure 2. Times-Roman Font Display of Word fine

Conversely, in some representations of the Spanish small letter enye (ñ), the base character and the tilde are encoded separately. This is unnecessary for Spanish if ISO-8859-1 or Unicode is used, but the facility is provided in Unicode. It is frequently useful in mathematics, where arbitrary meanings may be assigned to typographical accents. An example of a font which does not have semantic meaning in any human's native language is the standard X cursor font (see Figure 3).

Figure 3. X Cursor Font

An *encoding* is a mapping from each abstract character or glyph to one or more octets. For the common encodings of character sets and fonts for Western languages, only one octet is used. However, Asian languages have repertoires of thousands of ideographic characters; normally, two octets are used per character and two per glyph. Two formats are used for such large encodings. The first is the *wide character* format, in which each character is represented by the same number of octets. Examples are the pure Japanese JIS encoding and Unicode, which use two octets per character, and the ISO-10646 UCS-4 encoding (a planned superset of Unicode) which uses four octets per character. This encoding is the index mapping for a character set or font.

Another format is the *multibyte character* in which different characters may be represented by different numbers of octets. One example is the packed Extended UNIX Code for Japanese (8-bit EUC-JP), in which ASCII characters are represented in one octet which does not have the eighth bit set, and Japanese characters are represented by two octets. These octets are the same as in the plain JIS encoding, except that the eighth bits are set (in pseudo-C code, **euc = jis | 0x8080**). By using this encoding, any 8-bit-clean compiler designed for ASCII can be used to compile programs which use Japanese in comments and strings. This option would not be available for wide-character formats. If programs were written in pure JIS, the compiler would have to be rewritten to accept JIS ASCII characters. The ASCII character set is a subset of the JIS character set, but instead of being assigned the range 0x00 to 0x7F, the letters and digits are assigned values given by the ASCII value + 0x2300, and punctuation is scattered with no such simple translation. Another common place to encounter multibyte characters is in transformation formats, specifically the file-system-safe transformation of Unicode, UTF-8. Like EUC-JP, UTF-8 encodes the ASCII characters as single bytes in their standard positions.

Multibyte formats do not interfere with handling of text where the program does not care about the content (operations such as concatenation, file I/O and character-by-character display), but will work poorly or inefficiently where the content is important and addressing a specific character position is necessary (operations such as string comparison, the basis of sorting and searching). They may be especially useful for backward compatibility with systems designed for and implemented under the constraints of ASCII, e.g., compilers. They may be more space-efficient if the single-octet characters are relatively frequent in the text.

Wide-character formats are best where addressing specific character positions is important. They cannot be backward compatible with systems designed for single-octet encodings, although with appropriate choice of encoding, e.g.,

Unicode, little effort beyond recompiling with the type of characters extended to the size of wide characters may be necessary. Unfortunately, existing standards for languages like C do not specify the size of a wide character, only that it is at least one byte. However, the most recently designed languages often specify Unicode as the internal encoding of characters, and most system libraries specify a wide-character type of two bytes, which is equivalent to two octets.

Text Input

Text input is in some senses the inverse of text display. But, because computers are much better at displaying graphics than reading them, it presents problems of its own.

Probably the easiest method of text input would be voice. However, voice-input technology is still in its infancy, and there are times when a direct textual representation is preferable, such as for mathematics and computer programming. A highly adaptive system could be created, but typed keyboard input will be faster and more accurate for some time. Similarly, although optical character recognition (OCR) and handwriting recognition are improving rapidly, keyboard input will also remain more efficient for large bodies of text. The difference between OCR and handwriting recognition is that OCR treats static two-dimensional data, whereas handwriting recognition has the advantage of dynamics; this is particularly important in recognizing handwritten ideographic characters in Oriental languages. However, both of these would be quite close to an exact inverse of text display; the system would map from the physical inputs to the internal encoding directly, without user intervention. The fact that voice and optical technologies are not available for Linux systems makes the point moot for the moment.

For practical purposes, most Linux systems are limited to keyboard input. Several problems are related to internationalizing keyboard input. The first is that most computer keyboards are well-designed for at most one language. U.S. computer keyboards are not well-adapted to produce characters in languages that use accents. But the obvious solution, which is to add keys for the accented characters, is inefficient for those languages which have many accented characters (Scandinavian) or context-dependent forms (Arabic). It is impossible for languages which use ideographic character sets such as Chinese Hanzi or complex syllabaries like Korean Hangul. What is to be done for languages such as Greek and Russian with their own alphabetic scripts that can conveniently be mapped on a keyboard, but which cannot be used for programming computers?

The solution is the creation of input methods which translate keystrokes into encoded text. For example, in GNU Emacs with Mule, the character “Latin small

letter u with umlaut” can be input on U.S. keyboards via the keystrokes **"u**. However, this presents the problem that the text **upper** cannot be directly input one keystroke per character. Furthermore, umlauts are not appropriate accents for consonants; even for vowels, languages vary as to which vowels may be accented with umlauts. So this usage of the keystroke **"** must be context dependent within the input stream and must be conditioned by the language environment.

In the case of ideographic Oriental languages, the process is even more complex. Of course, it is possible to simply memorize the encoding and directly input code points, e.g., in hexadecimal. It is more efficient for ISO-2022-compatible encodings to memorize the two-octet representation as a pair of ASCII characters. Although this method of input is very efficient, it takes intense effort and a lot of time to memorize a useful set of characters. Educated Japanese adults know about 10,000; Unicode has 20,902. Furthermore, if you need a rare character that you have not memorized, the dictionary lookup is very expensive, since it must be done by hand. For these languages, the most popular methods involve inputting a phonetic transcription which the input method then looks up in an internal dictionary. The function which accepts keystrokes, produces the encoded phonetic transcription and queries the dictionary is often called a *front-end processor*, while the dictionary lookup is often implemented as a separate server process called the *back end*, *dictionary server* or *translation server*.

Dictionary servers often define a complex protocol for refining searches. In Japanese, some ideographic characters have dozens of pronunciations and some syllables correspond to over 100 different characters. The input method must weed out candidates using context, in terms of characters that are juxtaposed in dictionary words and by using syntactic clues. Even so, it is not uncommon that rather sophisticated input methods will produce dozens of candidates for a given string of syllables. Japanese has many homonyms, often with syntactically identical usage; occasionally, even with the help of context, the reader must trust that the author has selected the right characters. An amusing example occurred recently in a church bulletin, where the Japanese word “megumi”, meaning “(God's) grace”, was transcribed into a pair of characters that could easily be interpreted as a suffix meaning “gang of rascals”. The grammatical usage was different from the noun “megumi”, but as it happened, it would have been acceptable in the context of that issue. Only the broader context of the church bulletin made the typographical error obvious.

Obviously, substantial user interaction is necessary. Most input methods for Japanese involve presenting the user with a menu of choices; however, the interaction goes beyond this. The input methods will give the user a means to register new words in the dictionary and often a way to specify the priority in

candidate lists. Furthermore, dictionaries are pre-sorted according to common usage, but sophisticated input methods will keep track of each user's own style, presenting the candidates used most often early in the menu.

Users often have preferences among input methods even for the relatively simple case of accented characters in European languages, so each user will want to make the choice himself. Furthermore, no current input method is useful for more than two or three languages. Wnn, a dictionary server originally developed for Japanese, also handles Chinese and Korean with the same algorithms, although each language is served by a separate executable. The implication for internationalization is that protocols for communicating between applications and input methods will be very useful, so that users may select their own favorite and even change methods on the fly if the language environment changes. In X11R6, this protocol is provided by the X Input Method (XIM) standard; however, no such protocol is currently available for the console.

Although detailed discussion of the input methods themselves is beyond the scope of this article, I will describe the most common approaches to user interfaces for input methods. First of all, for non-Latin alphabetic scripts, the keyboard will simply be *remapped* to produce appropriate encoded characters. Both X and the Linux console provide straightforward methods for doing this. For novice users, the key-caps will need to be relabelled as well; touch-typists won't even need that.

For accented scripts, unless the number of accented characters is very small, it will not be possible to assign each one to its own key. One method of handling accents is the *compose key*, a special key which does not produce an encoded character itself but introduces a sequence of keystrokes which are interpreted as an accented character. Compose key methods typically need not be invoked or turned off by the user; they are simply active all the time. Since a special key is used, they do not interfere with the native language of the keyboard. The accent may be given a key of its own, but commonly some mnemonic punctuation mark is used, e.g., the apostrophe is mapped to the acute accent.

An alternative to the compose key is the *dead-key method*. Certain keys are called dead keys because they do not produce encoded characters; instead, they modify a contiguous character by placing an accent on it. Dead-key methods can be either *prefix* methods or *postfix* methods, depending on whether the modifier is entered before or after the base character. Obviously, these methods do interfere with input in other languages; a means of toggling them on and off is necessary.

Compose key methods are analogous to the use of a shift key to capitalize a single letter; dead-key methods are like the use of shift lock. Which is better depends on user preference and the task. Keyboard remapping, combined with either the compose key or the dead-key method, is sufficient to handle all of the ISO 8859 family of character sets.

Text Processing and Locales

Processing text is an extremely complex and diverse field of application. Currently, most aspects of localization and therefore internationalization have to be handled by each application according to its own needs. Programmers who want their applications to have the broadest possible utility should pay attention to internationalization issues, using standard techniques such as gettext wherever possible. They should avoid optimizations, such as using high bits in bytes or unused code points in an encoding for non-character information, that might conflict with extension to a new character set. Where standards have not yet evolved, internationalization demands that programmers design their own protocols for application-specific functionality that needs localization. Wherever possible, complex operations on text should be localized to a few functions that can be generalized to other languages or made conformable if a new standard is adopted.

Many areas come up which have been standardized already: numeric formatting, date formatting, monetary formatting and sorting. Yes/no answers have also been standardized, but this is superseded by GNU gettext.

Each of these functions has one or more functions provided by the POSIX standard for the C standard library. Linux's libc has not historically provided them, but they are all more or less fully provided in version 2 of the GNU C library. These functions are controlled by **locale**, an environmental parameter encoding various cultural aspects of text processing.

Locale is explicitly set in a program using the **setlocale** function. The current locale can be retrieved using the same setlocale function. Internally, each locale is divided into several parts which can be controlled separately. Users normally inform programs about their locale preferences using one or more environment variables (LANG, LC_ALL, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, LC_MESSAGES).

First of all, the convention for naming locales indicates the language, the regional subvariant and the encoding used. The “portable locale” is an exception; it has two names, C and POSIX. They have identical meaning. Two-letter abbreviations for language (ISO 639) and country (ISO 3166) have been standardized. So U.S. English, for example, may be specified as en_US.iso646-irv. (ISO 646-IRV is the version of the U.S. ASCII standard published by the ISO.)

This has a slightly different meaning from en_US.iso8859-1, in that the latter specifies use of the ISO 8859-1 encoding, which would permit the direct inclusion of accented letters from German or French if the necessary fonts were available. The former does not. How these differences are handled will be implementation-dependent, and even within Linux the console driver handles this differently from X.

British English would be specified as en_GB.iso8859-1. (ISO 646-IRV would not be satisfactory here, as it does not include the pound currency sign.) The U.S. and British locales do not differ on things like spelling. Theoretically, **ispell** could take a hint from the LANG variable, but as far as I know it does not. The main differences would be currency formatting and of course the dates, i.e., the U.S. uses MM/DD/YY, while Britain uses DD/MM/YY. Furthermore, Linux provides an English language locale for Denmark (reflecting the nationality of Keld Simonsen, who coordinates the locale library for WG15 of the ISO); this locale uses the Danish kroner as currency unit and the ISO 8601 standard YY/MM/DD for dates.

Another example is the several Japanese locales. While the Japanese language is widely used only in Japan, so that all of them start with ja_JP, there are several commonly used encodings. The locale normally used on Japanese Linux systems is ja_JP.eucJP (using the EUC-JP encoding), but internationalized software running on Japanese MS Windows systems would presumably use the ja_JP.sjis locale (using Microsoft's Shift-JIS encoding for Japanese). The reason for the difference is that UNIX file systems are compatible with file names encoded in EUC-JP, while MS Windows file systems will use file names in the somewhat different Shift-JIS encoding. This could cause problems with Japanese file names in MS-DOS and VFAT file systems mounted on a Linux system, as the POSIX locale system does not allow for multiple locales to be simultaneously active. Some elderly Japanese systems which are not 8-bit clean might use the ja_JP.jis locale, with the basic seven-bit JIS encoding.

The following aspects of a system are affected by the locale. The first five are implemented in libc according to the POSIX standard. Others are implemented in X or *ad hoc* by application software. Those implemented by libc are controlled explicitly by the setlocale call, which normally will default to the contents of the environment variable LANG. The remainder are implemented as “advice” encoded in the LANG environment variable or other environment variables.

- numeric formatting
- date formatting
- monetary formatting
- collating (sorting)

- yes/no messages
- display fonts
- file system encoding
- text file encoding
- input method

The display fonts are normally set by the application according to the encoding portion of the locale (after the period). Russian, Japanese and traditional Chinese all have multiple encodings. However, most fonts are provided in only one encoding, so applications must re-map other encodings internally. If this remapping is not done properly, the display will be unintelligible “mojibake”, pronounced MOH-JEE-BAH-KAY, a Japanese word literally meaning “changed characters” but more fancifully translated “monster characters”. Compare the Japanese text encoded in EUC-JP in Figure 4 with the mishmash of punctuation and funny characters produced when the kterm is explicitly told to interpret the text as Shift JIS in Figure 5. This is familiar to users of programs ported from DOS which use text windows based on the PC line-drawing characters. Since most Linux fonts are based on the ISO-8859-1 character set, you get “French windows” bordered in frilly accented characters rather than lines.

Figure 4. Japanese Text Encoded in EUC-JP

Figure 5. Japanese Text Encoded in Shift JIS

The file system encoding is also retrieved from the **advice** in the encoding portion of the LANG variable. Here, it is critical that the application be very defensively programmed. Carelessly accepting the advice may result in files with names which get corrupted when inserted in the file system or which cannot be accessed.

Text file encodings are similarly defaulted to the advice in the encoding portion of the LANG variable. However, in a networked environment, alien files will often be imported, e.g., using FTP, and there is no reason to suppose that these files will have the same encoding as the current locale. Applications should provide a means of specifying the encodings of files used; in locales where multiple encodings are available (today, Japanese, Russian and traditional Chinese, but soon with the popularization of Unicode and UTF-8, all locales), utilities for translating among compatible encodings should be provided.

Finally, a default input method can often be guessed from the LANG variable. In current systems, it will often be bundled with the keyboard mapping or console driver. X provides a more flexible system in which the XMODIFIERS environment variable is consulted to learn the user's preferred input method for each locale.

Next month, I will look at the large body of internationalization standards which have evolved to handle these problems.

Resources

Stephen Turnbull (turnbull@sk.tsukuba.ac.jp) is an economist teaching and researching in Japan. He is excited about the way the Open Source movement is turning conventional economics on its head, but is too busy playing with his Linux systems to do much economic analysis.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Autonomous Vehicles

Massimo Bertozzi

Alberto Broggi

Alessandra Fascioli

Issue #59, March 1999

Linux drives the experimental vehicle of the University of Parma.

The goal of the ARGO Project is to develop active safety systems for vehicles of the future. These systems are able to understand the environmental conditions and, in case of sudden danger, they can warn the driver or even take control of the vehicle. Furthermore, these systems have the capability to fully drive the vehicle without human intervention (Automatic Driving).

Figure 1. The ARGO Prototype Vehicle

ARGO (see Figure 1), the prototype vehicle developed at the Università di Parma, Italy, was demonstrated to the public and the scientific community in June 1998, when it drove autonomously for more than 2000km on the Italian highway network.

The entire real-time data acquisition and processing is performed on a single Pentium MMX-based PC running Linux.

Motivation

A large number of problems (not just those related to traffic and safety) could be solved by using automatically driven vehicles.

Beside the obvious advantages of increasing safety and reducing road accidents, thereby saving lives, the possibility of having vehicles moving in much closer proximity than they do today would produce an increase in road capacity. A more intelligent modulation of each vehicle's speed would also result in an appreciable reduction in fuel consumption. In other words,

automatic vehicle driving has the potential to achieve optimal use of current transportation infrastructures, improve mobility and minimize risks, travel times and energy consumption. Moreover, commercial and industrial vehicles which repeatedly move along a given path would benefit from a stronger control of their routes and would require fewer personnel to manage their moves.

Unfortunately, an automatic vehicle, more than in other applications, needs real-time response, thus requiring smart algorithms and powerful computing engines. At the same time, as far as a commercial product is concerned, the cost of the system must be kept low. Initially, the underlying technology, such as head-up displays, infrared cameras, radars and sonars are derived from expensive military applications. Thanks to increased interest in these applications and the progress of industrial production, today's technology offers sensors, processing systems and output devices at very competitive prices.

This project shows how a very low-cost solution (regarding both sensors and processing engines) was successfully adapted to drive an “intelligent” vehicle in real-world conditions. This article describes the underlying architecture, which is based on a standard Pentium MMX 200 MHz and the Linux operating system, and discusses the main advantages and problems encountered during the whole project.

The project was founded by the Italian National Research Council (CNR), and the MilleMiglia in Automatico tour was sponsored by TIM, Telecom Italia Mobile, which provided GSM apparatus and cellular connectivity throughout the whole journey.

System Architecture

The heart of the computing architecture installed on ARGO (shown in Figures 2 and 3) is based on a single Pentium MMX processor (200MHz, 32MB of RAM). The PC was equipped with some additional boards for image acquisition, image visualization, acoustic warnings and data I/O from specific devices.

Figure 2. Internal View of the ARGO Vehicle

Figure 3. The Computing Architecture and Equipment

GOLD (Generic Obstacle and Lane Detection), the software that drives the ARGO vehicle, has been designed to be as portable as possible; therefore, it is independent of specific Linux distributions/libraries and can be compiled against either libc or glibc. Moreover, because of the high stability of the first

kernel (2.0.18) we installed, the current distribution used on ARGO is still a Debian 1.2.

The following is a description of the ARGO adapters/software libraries as well as their support for Linux.

Acquisition device: since ARGO relies on a vision-based processing engine, the acquisition device is the most important piece of hardware. GOLD needs a frame grabber capable of simultaneously grabbing two grey-level images from two different cameras. Thanks to the availability of specific mailing lists about Linux and Vision (http://atlantek.com.au/USERS/wes/linux/frame.html), we discovered that only the Matrox Meteor RGB frame grabber had this feature among the Linux-supported hardware. The Matrox Meteor module is not yet video4linux compliant. Nevertheless, the availability of many examples as well as their source code permitted us to rapidly interact with this hardware.

Data I/O: while the frame grabber could be devised as the main input device, autonomous driving also requires an output device for maneuvering the steering wheel. A National Instruments LabPC+ ISA adapter has been installed in the ARGO PC; it provides a number of multi-functional analog, digital and timing I/O ports. Therefore, it is used for driving the steering wheel actuator (a stepping motor), determining the vehicle's speed (through a Hall-effect speedometer) and inputting user commands from the control panel as in Figure 2. The LabPC+ usage is quite simple, since the Linux module supplies a device and simple ioctls for each different I/O port.

Acoustic warnings: acoustical warnings are fed to the driver. A simple interface for a standard sound card has been developed using the OSS free sound module.

Figure 4. A Screenshot of the Output from the Onboard Monitor

Image visualization: the result of the computation is also fed to passengers through a color 6-inch monitor installed onboard the ARGO vehicle (see Figure 4). Development of the software requires the capability to both inspect intermediate results and input debug commands. For these purposes, two interfaces have been developed, one VGA-based and one X11-based. In the first case, the SVGA library is used for showing the (intermediate) results and the ncurses library is used to input user commands. In the second case, the xform widget library is also used (see Figure 5).

Figure 5. The X11-based Control Panel

Programming facilities: the MMX capabilities of the Pentium processor were exploited in order to boost processing speed. A few portions of the GOLD code were directly rewritten using MMX assembly language and compiled using the Netwide Assembler (NASM), a general-purpose x86 assembler that supports Pentium, P6 and MMX opcodes.

Internet connection: during the vehicle's demonstrations, live video shots of the driving cabin are broadcast to the Internet. To accomplish this, a link to the Internet was established. This type of link implies the use of mobile telecommunications facilities, such as GSM or satellite modems and places the following constraints:

- low bit-rate band for transmission (usually a little more than one kilobyte/second)
- high variability of the bandwidth during movements
- frequent carrier losses

In order to increase the link's throughput, two GSM modems are used simultaneously. In fact, the Linux kernel provides a specific method of making multiple serial links behave as a single faster connection: the EQualize Load balancer (EQL). The underlying idea is to split network traffic across the serial lines. In addition, the EQL also supports links that feature different throughputs and protocols.

In order to not excessively burden the main processing engine, another cheap Linux box (a Compaq laptop) was installed on the vehicle and was equipped with a parallel port Quickcam Color camera (supported by Linux) and two GSM modems able to work up to 9600Kbps using the MNP10-EC protocol. A custom application is used to grab images from the Quickcam, convert them to the JPEG format and send them to our web server at http://MilleMiglia.ce.unipr.it/ (a third Linux machine) through the two GSM modems exploiting EQL. As soon as these images are received, a graphical timestamp is superimposed onto their lower portion and they are made available on the Internet. A simple script running continuously in the background is used to restore the two connections if, for any reason (tunnels, GSM uncovered areas, etc.), they are lost.

On Road Test Journey

In order to test the vehicle extensively under different traffic conditions, road environments and weather, a 2000km journey was undertaken June 1 through June 6, 1998. During this test, ARGO drove autonomously along the Italian highway network, passing through flat areas and hilly regions including viaducts and tunnels. The Italian road network is particularly suited for such an extensive test since it is characterized by quickly varying road scenarios,

changing weather conditions and generally a fair amount of traffic. The tour took place on highways and freeways, but the system also proved to work on sufficiently structured rural roads with no intersections.

During the journey, in addition to the normal tasks of data acquisition and processing for automatic driving, the system logged the most significant data, such as speed, position of the steering wheel, lane changes, user interventions and commands, dumping the whole status of the system (images included) whenever the system had difficulties in detecting the road lane reliably.

This data was processed off-line after the end of the tour in order to compute the overall system performance, such as the percentage of automatic driving, and to analyze unexpected situations. The possibility of reprocessing the same images, starting from a given system status, allows the reproduction of conditions in which a fault was detected and a way of solving it found. At the end of the tour, the system log contained more than 1200MB of raw data, while during the whole tour the system processed about 1,500,000 images (each 768 x 288 pixels) totalling about 330GB of input data.

During the tour, the ARGO vehicle broadcast a live video stream to the Internet: two GSM cellular modems were connected to the Vision Laboratory of the Dipartimento di Ingegneria dell'Informazione in Parma, and were used to transfer both up-to-date news on the test's progression and images acquired by a camera installed in the driving cabin to demonstrate automatic driving. Proving there was high interest in the scientific community, mass media and general public, the web site http://MilleMiglia.CE.UniPR.IT/ was visited more than 350,000 times during the tour and more than 3000MB of information was transferred, with a peak of 16,000 visits per hour during the first day of the tour.

Performance Analysis

The main problem experienced during the tour was due to image acquisition. One aim of the project is the development of a sufficiently low-cost system that will ease its integration into a large number of vehicles, so the use of low-cost acquisition devices was a clear starting point. In particular, videophone cameras (small-sized sensors at an average cost of $100 US each) were installed. Although these cameras have a high sensitivity even in low-light conditions (at night), a sudden change in illumination of the scene (for example, at the entrance or exit from a tunnel) causes a degradation in the image quality. (They were designed for applications characterized by constant illumination, such as video-telephony.) The cameras have a slow automatic gain control to the exit from a tunnel for a period of about 100 to 200ms; thus, the acquired images are completely saturated and their analysis becomes impossible.

Figure 6. Automatic Driving During the MilleMiglia in Automatico Tour

On the other hand, the design of the processing system proved to be appropriate for automatic driving of the vehicle. Moreover, current technology provides processing systems with characteristics that are definitely more powerful than the one installed on ARGO: a commercial PC with a 200MHz Pentium processor and 32MB of memory. On such a system, enhanced by a video frame grabber able to acquire two stereo images simultaneously (with 768x576 pixel resolution), the GOLD system processes up to 25 pairs of stereo frames per second and provides the control signals for autonomous steering every 40ms. (This is equivalent to one refinement on the steering wheel position for every meter when the vehicle is driving at 100kmph.) Obviously, the processing speed influences the maximum safe vehicle speed: the higher the processing speed, the higher the maximum vehicle speed.

Differing weather conditions in particular light conditions demonstrated the robustness of both the approach and the image processing algorithms. In fact, the system was always able to extract the information for the navigation task even in critical light conditions, with the sun in front of the cameras, high or low on the horizon, during the evening as well as during the day, with high or low contrast. At night, the system's behavior is improved by the absence of sunlight reflections and shadows, while the area of interest is constantly illuminated by the vehicle headlights.

Figure 7. The MilleMiglia in Automatico Tour Route

Finally, the system proved to be surprisingly robust despite high temperatures measured during the tour. In some cases, the external temperature reached 35 degrees Celsius and the system continued to work reliably even with no air conditioning.

Statistical Analysis of the Tour

Table 1. System Performance Statistics

The analysis of the data collected during the tour allowed the computation of a number of statistics regarding system performance (see Table 1). In particular, for each stage of the tour, the average and maximum speeds of the vehicle during automatic driving were computed. Average speed was strongly influenced by the heavy traffic conditions (especially on Torino, Milano and Roma's bypasses) and by the presence of toll stations, junctions and road works.

The automatic driving percentage and maximum distance show high values, despite the presence of many tunnels (particularly during the Appennines

routes Ancona to Roma and Firenze to Bologna) and of several stretches of road with absent or worn lane markings (Ferrara to Ancona and Ancona to Roma) or even no lane markings at all (Firenze to Bologna). It is fundamentally important to note that some stages included passing through toll stations and transiting bypasses with heavy traffic and frequent queues, during which the system had to be switched off.

Closing Remarks

Throughout the entire project, the choice of an Intel-based platform, coupled with the Linux operating system, was shown to be extremely reliable; the number of faults due to these system components was zero over the last two years.

Originally, the main reason behind the choice of the Linux operating system (instead of real-time OS or operating systems for industrial PCs) was the availability of up-to-date developing and debugging tools, drivers and FAQs about specific hardware devices, and the possibility of interacting with a large number of researchers worldwide on the Internet in order to solve problems.

The main topics for future research (ARGO Project, phase II) are related to the development of a new vehicle integrating both road following and platooning (the automatic following of a manually driven vehicle) functionalities. In this next phase, the processing engine will be a higher performance Intel-based architecture, again driven by the Linux operating system.

Resources

Comments From Others about ARGO

Massimo Bertozzi holds a Ph.D. in Information Technology and a Master's degree in Electronic Engineering. He has been a Linux enthusiast since kernel 1.0.9. Since 1994, he has been working at the Università di Parma on research topics regarding machine vision and clusters of Linux boxes. When not recompiling strange programs, reading news, surfing the net, playing Quake or some other adventure game, he can be reached via e-mail at bertozzi@ce.unipr.it.

Alberto Broggi is married to Simona, who gently allows him to work at home and even during the night on topics related to automatic vehicle guidance using artificial vision. In his spare time, he teaches a number of courses at both Università di Pavia and Università di Parma and has still not learned to turn when someone calls him Professor. He is involved in the organization of many international scientific events, but had a bad time when looking for a photo with a tie to include in this biography. Alberto will certainly answer if you contact him at broggi@dis.unipv.it.

Alessandra Fascioli received the Dr.Eng. (Master) degree in Electronic Engineering from the Università di Parma, Italy. Currently, she is a Ph.D. candidate in Information Technology at the Dipartimento di Ingegneria dell'Informazione of the Università di Parma. Her research activities on real-time computer vision for automatic vehicle guidance and on image processing techniques based on the Mathematical Morphology computational model generally produce better results after a good swim or sunbath at the university campus swimming pool. You can contact her at fascal@ce.unipr.it.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Internationalizing Messages in Linux Programs

Pancrazio de Mauro

Issue #59, March 1999

An introduction to the GNU gettext system for producing multilingual programs.

Linux is becoming increasingly popular each day. Until now, the typical Linux user has been a system administrator, student or UNIX hacker. New projects such as GNOME, KDE and GNUStep are preparing the way for a different, less technically prepared user.

Running software in English is usually not a problem for someone with at least moderate computer skills, but end users need (and *want*) software that speaks their own language in order to be productive or feel comfortable with the system. Moreover, many programs need to know local conventions for things such as dates or money amounts in order to be useful and complete.

This article is an introduction to the *GNU gettext* system, a set of tools and libraries for both programmers and translators that enables them to produce multilingual programs with textual messages in specified languages. We will deal with languages that use one of the ISO-8859-X character sets, except for Japanese and Chinese as they require extra care.

Definitions

Two words appear frequently when talking about support of different languages in programs: *internationalization* and *localization*. Since writing these words over and over (without spelling errors) is annoying and time-consuming, people abbreviate them as *I18N* and *L10N*. The 18 and 10 indicate the number of letters between the first and the last letter of each word.

Internationalizing a program means taking the necessary steps to make it aware of different languages and national standards.

The process of *localization* takes place when an internationalized program is given the information needed to behave correctly with a certain language and set of cultural habits.

First Things First

The first thing to do, for both programmers and end users, is configure the Linux machine to use locales. Most users need only follow the *Locales mini-HOWTO* downloadable from ftp://sunsite.unc.edu/pub/Linux/docs/ and mirrors. Recent distributions (for example, Red Hat 5.0) include everything to support locales.

Once the system is enabled to support locales, you must specify the particular standards and languages you wish to use. This is done through a set of environment variables. Each one controls a specific aspect of the locale system:

- *LANG* specifies the global locale, but can be overridden by the following variables.
- *LC_COLLATE* specifies the locale used for sorting and comparing.
- *LC_CTYPE* specifies the character set in use, so that **isupper('<\#192>')** returns *true* in an Italian locale.
- *LC_MONETARY* provides information about representing money in a specific locale.
- *LC_NUMERIC* gives information about numbers: how digits are divided and separated in groups, what the decimal point is, etc.
- *LC_TIME* specifies which locale to use to represent time: AM/PM or 24-hour values, for example.
- *LC_MESSAGES* indicates the language you prefer for programs' text messages.
- *LC_ALL* overrides any previous indication and sets a global locale.

Examples of values for global locale are:

- **en_US** indicates English in the United States.
- **it_IT** is for Italian in Italy.
- **fr_CA** is for French in Canada.

Basically, to use the standards of language *LL* in country *CC*, the locale value will be **LL_CC**.

Listing 1.

The locale used by default, unless overridden by the previous variables, is called the **C** (or **POSIX**) locale. Thus, it is very easy to illustrate the behavior of a locale-aware program by using **date**, for example (see Listing 1). First, without setting the LC_ALL variable, the response is in English. Next, LC_ALL is set to obtain an Italian response, a French one (French in Canada is specified), then an English one (English in Canada). The “No such file or directory” for the Italian locale is not translated, which means the Italian information is not available; therefore, the default is used instead.

Dealing with Messages in C Programs

Let's have a first look at the package GNU gettext. If you don't have it installed on your system, you can download it from ftp://prep.ai.mit.edu/pub/gnu/ or its mirrors.

When writing multilingual programs with this package, strings are “wrapped” in a function call instead of being coded directly in the source. The function is called **gettext** and accepts exactly one string argument and returns a string.

Despite its simplicity, gettext is very effective: the string passed as an argument is looked up in a table to find a corresponding translation. If a translation is found, then gettext returns it; otherwise, the passed string is returned and the program will continue to use a default language.

Our first, internationalized *Hello, world!* program could be:

```
#include <stdio.h>
#include <libintl.h>
void main(void) {
        textdomain("hello-world");
        printf(gettext("Hello, world!\n"));
}
```

Always remember to include **<libintl.h>** in each C program that makes use of the gettext package.

The function **textdomain** should be called before using gettext. Its purpose is to select the correct “database” of messages (a more appropriate term would be “message catalog”) for the program to use.

Then, each translatable string must be used as a parameter of gettext. Writing **gettext("foobar")** each time can be annoying. That's why many programmers use this macro:

```
#define _(x) gettext(x)
```

By doing so, the overhead introduced by internationalization of messages is quite small: instead of writing **"foobar"**, one can just write **_("foobar")**. That's

only three characters more per translatable string, with the advantage that this macro eliminates the gettext code from the module completely.

Translating Messages

Once a program has been internationalized, the localization process can begin. The first thing to do is extract all the strings needing translation from the source code.

This automatic process is carried out by **xgettext**. The result is an editable .po (portable object) file. **xgettext** scans the source files passed as parameters and extracts each translatable string marked by the programmer with gettext or some other identifier.

Listing 2.

In our case, we can invoke **xgettext** in this way:

```
xgettext -a -d hello-world -k_ -s
-v hello-world.c
```

The resulting hello-world.po is shown in Listing 2.

I suggest you take a look at the gettext info documentation to learn about other useful switches. The ones I used here are defined in this way:

- **-a** extracts all strings.
- **-d** outputs the results in hello-world.po (the default is messages.po).
- -k instructs xgettext to look for **_** when searching translatable strings (the defaults **gettext** and **gettext_noop** are still looked for).
- **-s** generates a sorted output and removes duplicates.
- **-v** tells xgettext to be verbose when it generates messages.

At this point, the translator can simply fill hello-world.po with the messages without any knowledge of the source code. In fact, a program can be internationalized and compiled, before adding the new languages.

A portable object must be compiled into a machine object (a .mo file) to be useful. This is done with the command:

```
msgfmt -o hello-world.mo -v hello-world.po
```

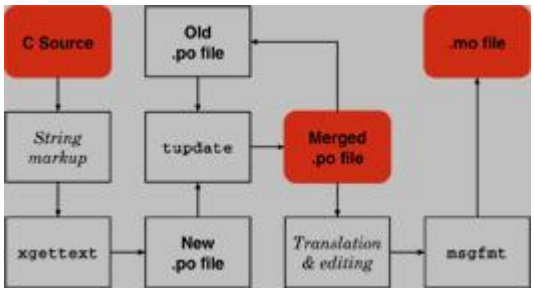


Figure 1

Figure 1. A block diagram representing all the steps necessary to obtain a **.mo** file from a C source. The most critical part is running **tupdate** (see below) to merge the new, untranslated strings with the previous work without losing it.

The final step is copying **hello-world.mo** to a suitable location, where it can be found by the gettext system. On my Linux box, the default location is /usr/share/locale/*LL*/ LC_MESSAGES/ or /usr/share/locale/*LL_CC*/LC_MESSAGES/, where *LL* is the language and *CC* is the country. For example, the Italian translation should be placed in /usr/share/locale/it/ LC_MESSAGES/hello-world.mo.

textdomain must be called in the beginning of the program, so that the system can select the proper **.mo** file according to the current locale variables. In order of precedence (higher precedence first), they are LC_ALL, LC_MESSAGES and LANG.

A .mo file can be shared among many programs if the programmers decide to make it so. This is true with GNU fileutils, for example.

Maintaining the Message File

If the source code changes, the corresponding .po file should be updated without losing any previous translation. Unfortunately, simply calling xgettext again does not work because it overwrites the old .po file. In this case, the program **tupdate** comes in handy. It merges two .po files, keeping translations already made, as long as the new strings match with the old. Its syntax is simple:

```
tupdate new.po old.po > latest.po
```

New strings will obviously still be empty in latest.po, but already translated ones will be there without the need for reprocessing.

Exceptions

Listing 3

It is not always possible to use the gettext function “straight”. Let's look at the source code excerpt in Listing 3 as an example. Two goals must be reached during the internationalization of this code. First, each translatable string must appear in the .po file. Second, before printing each string at runtime, we must pass it through gettext.

The string **"You have %d %s"** poses a problem. We cannot simply transform each string declared in **item_names** in a **gettext** call, because arrays must be initialized with constant values.

Listing 4

One solution is shown in Listing 4. **gettext_noop** is a marker used to make the string recognizable by xgettext (that is why it is looked for by default). The translation occurs at run time with the normal gettext call.

Message File Format

The .po files have a very simple text structure and can be modified with any text editor. Among others, Emacs can be put in a special *po* mode when dealing with them.

Each message file consists of a sequence of records. Each record has this structure:

```
(blank lines)
#  optional human comments
#. optional automatic comments
#: optional source code reference
msgid original-string
msgstr translated-string
```

Comments introduced by the translator should have a whitespace immediately following the **#** character. Automatic comments are produced by xgettext and tupdate to enhance the file's readability and to allow the translator to quickly browse the source code and find the line where a string is used. This is sometimes necessary to produce a correct translation.

Strings are formatted just like C. For example, it is legal to write:

```
msgid ""
"Hello     "
"world!\n"
msgstr ""
"Ciao      "
"mondo!\n"
```

As you can see, strings may span across lines and the backslash is used to introduce special characters such as tabs and newlines.

Other Message Catalog Systems

No POSIX standard for message catalogs exists—the committee could not agree on anything.

GNU gettext is not the only message catalog system that can be used by an internationalized program. Another library, based on the **catgets** function call, also exists. The catgets interface is supported by the X/Open consortium, while the gettext interface was first used by Sun.

The main disadvantage of catgets is that a *unique* identifier must be chosen for each message and passed to catgets each time. This makes it quite difficult to manage a large set of messages, where entries are inserted and deleted on a regular basis. However, GNU gettext can use catgets as an underlying interface on systems that support it.

Linux supports both gettext and catgets interfaces. My personal opinion is the gettext system is much easier to use for both programmers and translators.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue59/3023.tgz

Pancrazio de Mauro (pdemauro@datanord.it) is a technical writer and a Linux consultant. He spends most of his time advocating Linux and trying to convince his friends to call him Ezio which, of course, sounds as bad as Pancrazio in English.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Mediated Reality: University of Toronto RWM Project

Dr.. Steve Mann

Issue #59, March 1999

Dr. Mann describes his WearComp ("Wearable Computer") invention as a tool for "Mediated Reality". WearComp originated in the context of photographic tools as true extensions of the mind and body and evolved into a philosophical basis for self-determination, characteristic of the Linux operating system that runs on WearComp.

As I wrote last month, I am an inventor who likes to think outside the box and chose to use the Linux operating system because it gives me a programming environment in which the box is not welded shut. I described my framework for human-machine intelligence which I call Humanistic Intelligence (HI). I also described the apparatus of an invention I call "WearComp" that embodies HI. In particular, I outlined some of the reasons for choosing Linux for WearComp and emphasized how problems like that of software fascism are much more pronounced on WearComp than they are in the context of regular desktop computing.

In this article, I will explore practical uses of WearComp. I will also explain how WearComp turns the traditional business model of real-world information space (e.g., advertising) upside down.

Summarizing briefly, WearComp is a wearable computational device that provides the user with a self-created personal space. The most fundamental issue of WearComp is that of personal empowerment (see Resources 1). I will show by way of example how WearComp provides the wearer with a self-created visual space. I will also describe the concept of "Mediated Reality" and the use of a "visual filter" that allows the wearer to create a visual attention access control system.

Problem Statement

If the eye is the window to the soul, then our soul is available for anyone to steal. Our visual attention is a valuable resource that can be consumed by billboards, distracting advertising and other material often thrust upon us against our will.

Solitude is a valuable form of humanistic property all too easily subject to theft.

I am taking the liberty of using these strongly judgmental words—steal and theft. Such strong wording, however, is already present in the context of intellectual property. We readily accept terms like “software piracy”, which make an analogy between someone who copies a floppy disk and someone who seizes control of an ocean-going vessel, often killing everyone on board. An analogy between such gruesome mass murder and copying software ought to raise certain questions about our social value system. Thus, against this backdrop, I believe that use of terms like theft and steal are not out of line in the context of what I call humanistic property.

Those who steal our solitude not only take away humanistic property, but force material upon us that can put our lives in danger.

Advertising is an evolving entity. In the old days, there were fixed signs with a static display of company slogans. Once we became accustomed to these signs, new ones were invented with more distracting and vibrant colours and even moving parts to fight for our attention. As we became accustomed to these signs, they were made brighter. Concepts such as light chasers, lamp sequencers and the like were introduced so that motion arising from sequentially illuminated bulbs could further distract us.

Then came the pixel boards, which also got brighter as we became accustomed to them. Some pixel boards use as many as 2000 watts per pixel. When lights this bright are put along major highways, they pose a serious threat to road safety. Still, we do our best to ignore these distractions and keep our eyes on the road or on whatever task has our attention.

The latest trend is something I call “signal-band advertising”. It tries to trick us by resituating advertising (“noise”) into what we perceive to be a “signal” band. For example, we are now seeing WWW banner advertisements with an imitation of a cursor; thus, the user is momentarily tricked into thinking there are two cursors on his screen. The advertisement contains what looks like a cursor, which moves around very much like a real cursor normally does. These kinds of ads are the cyberspace equivalent of trying to get attention by yelling “Fire!” in a crowded movie theater.

Another example of signal-band advertising exists on parking-lot booms. By renting a sign on the boom of a parking lot, advertisers can further confuse drivers by placing ads where only road signs would normally be. We now need to distinguish between advertising and important road signs, both of which are directly in our path in the center of the road. The advertising is no longer only off to the side of the road. This theft of visual attention makes it that much harder to see stop signs and other important traffic markers.

Perhaps next, advertisers will start to make their signs red and octagon-shaped and hang them on lamp posts along the street, so they will be able to grab even more of our attention. A red octagon, with a product slogan in white letters in the center, posted at a busy intersection could get lots of attention and would be harder to ignore than traditional billboards. This is what I mean by "signal-band advertising".

Those who steal our visual attention are not content to just clutter roads and open public space with advertising, but they appear to also want to intrude on more private spaces as well.

Mediated Reality

A solution to this problem may be obtained through something I call "Mediated Reality". Mediated Reality (MR) differs from virtual reality (or augmented reality) in the sense that it allows us to filter out things we do not wish to have thrust upon us against our will. This capability is implicit in the notion of self-determination and mastery over one's own destiny. Just as a Sony Walkman allows us to drown out Muzak with our own choice of music, MR allows us to implement a "visual filter". I will now describe how MR works. Later, we will see the importance of a good software basis for MR and why Linux was selected as the operating system for the apparatus of the invention (WearComp) upon which MR is based.

Operational Principle of Mediated Reality

Figure 1

To understand how the reality mediator works, imagine first a device called a "Lightspace Analyzer" (see Figure 1). The Lightspace Analyzer is a hypothetical "lightspace glass" that absorbs and quantifies incoming light—it is completely opaque. It provides a numerical description (e.g., it turns light into numbers). It is not necessarily flat; the analyzer is drawn curved in the figure to emphasize this point.

Figure 2

Imagine also a “Lightspace Synthesizer” (see Figure 2). The Lightspace Synthesizer turns an input stream of numbers into corresponding rays of light.

Figure 3

Suppose we connect the output of the Lightspace Analyzer to the input of the Lightspace Synthesizer (see Figure 3). We now have an illusory transparency.

Moreover, suppose we could bring the Lightspace Analyzer glass into direct contact with the Lightspace Synthesizer glass. Placing the two back to back would create a collinear illusory transparency, in which any emergent ray of virtual light would be collinear with the incoming ray of real light that gave rise to it. (See Figure 4.)

Figure 4

Now, a natural question to ask is, why all this effort in making a simple illusion of transparency when we could just as easily purchase a small piece of clear glass? The answer is that we have the ability to modify our perception of visual reality by inserting a WearComp between the Lightspace Analyzer and the Lightspace Synthesizer. (See Figure 5.)

Figure 5

In practice, there are other more practical embodiments of this invention than the one described above, but the basic principle is the same. Some practical examples are described further elsewhere in the literature (see *Proceedings of IEEE ISWC98*, “WearCam, the Wearable Camera”, by Steve Mann, pages 124-131). The result is a computational means of altering one's visual perception of reality.

All the World's a Web

WearComp has the potential to make all the world virtual as well as real; moreover, the potential is there to create a modified perception of visual reality. Such a computer-mediated reality can not only augment, but also diminish or otherwise alter the perception of reality.

Why would one want to do this? Why would anyone buy a pair of sunglasses that made them see less?

An example might be when we are driving and trying to concentrate on the road. Sunglasses that not only diminish the glare of the sun's rays but also filter out distracting billboards could help us see the road better, and therefore drive more safely.

Figure 6. Sunglasses for Filtering

Moreover, Mediated Reality can help us reclaim solitude in personal spaces. By wearing special sunglasses in which a visual filter is implemented (see Figure 6), it is possible to filter out offensive advertising.

Figure 7. Top: Convention Hall Reality. Second from top through bottom: Convention Hall Mediated Reality

Lightspace entering the analysis side of the glass manifests itself as an input image sequence where it is absorbed and quantified by the special sunglasses. Figure 7 shows Convention Hall as it truly is; Figure 8 shows its transformation with visual filtering.

Recall that the sunglasses are totally opaque except for the fact that the WearComp copies the input side to the output side, with possible modification. In the case of an offensive advertisement, the modification could take the form of replacing the advertisement with a more calming image of a waterfall.

Connected Collective Human Intelligence

Personal Imaging is a camera-based computational framework in which the camera behaves as a true extension of the mind and body, after a period of long-term adaptation (see Resources 2). In this framework, the computer becomes a device that allows the wearer to augment, diminish, or otherwise alter his visual perception of reality. Moreover, it lets the wearer allow others to alter his visual perception of reality, thereby becoming a communication device.

The communication capabilities of WearComp allow for multiple wearers of the special sunglasses to share a common visual reality. Currently, the sunglasses are connected to the Internet by way of a 2Mbps (megabit per second) radio. This is a significant speed upgrade from the old 1987 radio design (running at only 56Kbps); thus, the shared realities may be updated at a much higher rate. The current system permits real-time video update rates for shared video.

Reality User Interface

One application of computer-mediated reality is to create, for each user of the apparatus, a possibly different interpretation of the same visual reality. Since the apparatus shares the same first-person perspective as the user (and in fact the apparatus is what enables the user to see at all), then, of course, the apparatus provides the processing system (WearComp) with a view of how the user is interacting with the world. In this way, each user may build his or her own user interface within the real world. For example, one user may decide to

have the computer automatically run a telephone directory program whenever it sees the user pick up a telephone. This example is similar to hypertext, in the sense that picking it up is like clicking on it with a mouse as if it were in an HTML document. “Clicking” on real objects is done by simply touching them.

Outlining objects with the fingertip is another example of a reality user interface (RUI).

Reality Window Manager

When windows are used together with the RUI, a new kind of window manager results. For example, while waiting in a lounge or other waiting area, a user might define walls around the lounge as various windows. In this way, screen real estate is essentially infinite. Although not all screens are visible at any one time, portions of them become visible when they are looked at through the WearComp glasses. Others in the lounge need not be able to see them, unless they are wearing similar glasses and the user has permitted them access to these windows (as when two users are planning upon the same calendar space).

There are no specific boundaries in this form of window manager. For example, if a user runs out of space in the lounge, he or she can walk out into the hall and create more windows on the walls of the hallway leading into the lounge. It is also easier to remember where all the windows are when they are associated with the real world. Part of this ease of memory comes from having to walk around the space or at least turn one's head around in the space.

This window manager, called RWM, also provides a means of making the back of the head “transparent” in a sense so that one can see windows in the front as rightside up and windows behind as upside down. This scheme simply obeys the laws of projective geometry. Rearview windows may be turned on and off, since they are distracting for concentration, but they are useful for quick navigation around a room. An illustration depicting the function of RWM to operate a video recording system is given in Figure 8.

Figure 8

A vision analysis processor typically uses the output of the Lightspace Analyzer for head tracking. This head tracking determines the relative orientation (yaw, pitch and roll) of the head based on the visual location of objects in the Lightspace Analyzer's field of view.

A vision analysis processor is implemented in the WearComp, as well as remotely, by way of the radio connection. The choice of which of these to use is made automatically based on how good a radio connection can be established.

The vision analysis processor does 3-D object recognition and parameter estimation, or constructs a 3-D scene representation. An information processor takes this visual information and decides which virtual objects, if any, to insert into the Lightspace Synthesizer.

A graphics synthesis processor creates a computer-graphics rendering of a portion of the 3-D scene specified by the information processor and presents this computer-graphics rendering to the wearer by way of the Lightspace Synthesizer.

The objects displayed are synthetic (virtual) objects overlaid in the same position as some of the real objects from the scene. The virtual objects displayed on the Lightspace Synthesizer correspond to real objects within the Lightspace Synthesizer field of view. Thus, even though the Lightspace Synthesizer may only have 480 lines of resolution, a virtual television screen, of extremely high resolution, wrapping around the wearer, may be implemented by virtue of the Lightspace Analyzer head-tracker, so that the wearer may view very high-resolution pictures through what appears to be a small window that pans back and forth across the picture triggered by head movements of the wearer.

Optionally, in addition to overlaying synthetic objects on real objects to enhance them, the graphics synthesis processor may cause the display of other synthetic objects on the virtual television screen.

For example, Figure 9 illustrates a virtual television screen with some virtual (synthetic) objects such as an Emacs Buffer upon an xterm (text window in the commonly used X Window System graphical user interface). The graphics synthesis processor causes the Lightspace Synthesizer screen to display a reticle seen in a virtual view finder window.

The viewfinder has 640 pixels across and 480 down, which is just enough resolution to display one xterm window since an xterm window is typically 640 pixels across and 480 down also (sufficient for 24 rows of 80 characters of text). Thus, by turning his head to look back and forth, the wearer can position the viewfinder reticle on top of any number of xterms that appear to hover in space above various objects. The true objects, when positioned inside the mediation zone established by the viewfinder, may also be visually enhanced as seen through the viewfinder.

Figure 9.

Suppose the wearer of the apparatus is in a department store and, after picking up a $7 item for purchase, he hands the cashier a $20 dollar bill, but receives

only $3 change (e.g., receives change for a $10 bill). Upon realizing this fact a minute or so later, the wearer locates a fresh, available (e.g., one that has no programs running in it so that it can accept commands) xterm. The wearer makes this xterm active by head movement up and to the right, as shown in Figure 9. Thus, the Lightspace Analyzer (typically implemented by a camera with special optics) functions also as a head tracker, and it is by orienting the head (and hence the camera) that the cursor may be positioned. Making a window active in the X Window System is normally done by placing the mouse cursor on the window and sometimes clicking on it. However, using a mouse with a wearable camera/computer system is difficult, owing to the fact that it requires a great deal of dexterity to position a cursor while walking around. With the invention described here, the wearer's head is the mouse and the center of the viewfinder is the cursor.

In Figures 8 and 9, objects outside the viewfinder mediation zone are depicted in dashed lines, because they are not actually visible to the wearer. He can see real objects outside the field of vision of the viewfinder (either through the remaining eye, or because the viewfinder permits one to see around it). However, only xterms in the viewfinder are visible. Portions of the xterms within the viewfinder are shown with solid lines, as this is all the wearer will see.

Once the wearer selects the desired window by looking at it, he then presses “d” to begin “recorDing”, as indicated on the window selected. Note that “d” is pressed for “recorD”, because “r” means “Recall” (in some ways equivalent to “Rewind” on a VCR). Letters are selected by way of a small number of belt-mounted switches that can be operated with one hand, in a manner similar to what courtroom stenographers use to form letters of the alphabet by pressing various combinations of pushbutton switches. Note that the wearer does not need to look right into the center of the desired window: the window accepts commands as long as it is active and doesn't need to be completely visible to accept commands.

Recording is typically retroactive, in the sense that the wearable camera system, by default, always records into a 5-minute circular buffer, so that pressing “d” begins recording starting 5 minutes before “d” is actually pressed. This means that if the wearer presses “d” within a couple of minutes of realizing that the cashier shortchanged him, then the transaction will have been successfully recorded. The customer can then review the past 5 minutes and can assert with confidence (through perfect photographic/videographic memory Recall, e.g., by pressing “r”) to the cashier that a $20 bill was given. The extra degree of personal confidence afforded by the invention typically makes it unnecessary to actually present the video record (e.g., to a supervisor) in order to correct the situation. Of course, if necessary, the customer could file a report or notify authorities while at the same time submit the recording as

evidence. The recording is also sent to the Internet by way of the 2Mbps transmitter so that the cashier or other representatives of the department store (such as a security guard who might be a close friend of the cashier) cannot seize and destroy the storage medium upon which the recording was made.

Note that here, the drawings depict objects moved translationally (e.g., the group of translations specified by two scalar parameters), while in actual practice, virtual objects undergo a projective coordinate transformation in two dimensions governed by eight scalar parameters, or objects undergo three-dimensional coordinate transformations. When the virtual objects, such as text windows, are flat, the user interface is called a “Reality Window Manager”.

When using the invention, various windows appear to hover above various real objects. Regardless of the orientation of the wearer's head (position of the viewfinder), the system sustains the illusion that the virtual objects (in this example, xterms) are attached to real objects. Panning the head back and forth in order to navigate around the space of virtual objects may also cause an extremely high resolution picture to be acquired through appropriate processing of multiple pictures captured on the special camera. This action mimics the function of the human eye, where saccades are replaced with head movements to sweep out the scene using the camera's light measurement ability, typical in “Photoquantigraphic Imaging”. Thus, head movements are used to direct the camera to scan out a scene in the same way eyeball movements normally orient the eye for this purpose.

The VideoOrbits Head Tracker

Of course, one cannot expect a head-tracking device to be provided in all possible environments, so head tracking is done by the reality mediator, using the VideoOrbits (see Resources 3) tracking algorithm. (The VideoOrbits package upon which RWM is based is freely available at http://wearcam.org/orbits/index.html.) The VideoOrbits head tracker does head tracking based on a visually observed environment, yet works without the need for high-level object recognition.

VideoOrbits builds upon the tradition of image processing (see Resources 4 and 5) combined with the Horn and Schunk equations (see Resources 6) and some new ideas in algebraic projective geometry and homometric imaging, using a spatiotonal model, p, that works in the neighborhood of the identity:

Figure 10

where $\mathbf{\emptyset T = [Fx(xy, x, y, 1), Fy(xy, x, y, 1), F, 1], F(x,t) = f(q(x))}$ at time t, $\mathbf{Fx(x,t) = (df/dq)(dq(x)/dx)}$, at time t, and $Ft(x,t)$ is the difference of adjacent

frames. This “approximate model” is used in the innermost loop of a repetitive process, then related to the parameters of an exact projectivity and gain group of transformations, so that the true group structure is preserved throughout. In this way, virtual objects inserted into the “reality stream” of the person wearing the glasses, follow the orbit of this group of transformations, hence the name VideoOrbits.

A quantagraphic version of VideoOrbits is also based on the fact that the unknown nonlinearity of the camera, *f*, can be obtained from differently exposed images *f*(*q*) and *f*(*kq*), etc., and that these can be combined to estimate the actual quantity of light entering the imaging system:

Figure 11

where *ci* is the derivative of the recovered nonlinear response function of the camera, *f*, and *A*, *b* and *c* are the parameters of the true projective coordinate transformation of the light falling on the image sensor. This method allows the actual quantity of light entering the reality mediator to be determined. In this way, the reality mediator absorbs and truly quantifies the rays of light entering it. Moreover, light rays entering the eye due to the real and virtual objects are placed on an equal footing.

The New Business Opportunity in Mediated Reality

MR sets forth a new computational framework in which the visual interpretation of reality is finely customized to the needs of each individual wearer of the apparatus. The computer becomes very much like a prosthetic device or like prescription eyeglasses. Just as you would not want to wear undergarments or another person's mouth guard, you may not want to find yourself wearing another person's computer.

The traditional paradigm of one worldwide software vendor providing everyone with identical copies of an executable-only distribution no longer applies. Instead, complete reconfigurability is needed and each user will customize his or her own environment. Since many laypersons are not well-versed in operating system, kernel source code, a need will grow for system administrators and consultants.

In the future, software will be free and users will buy support. There will be little problem with software piracy, both because the software will be free and because a version of the software customized for one person will be of less use to someone with different needs. Because the computer will function as a true extension of the user's mind and body, it would not do the user well to ingest software owned by someone else. The computer will function much like a

“second brain”, and in the true spirit of freedom of thought, it would be preferable that any commercial interests in the customization and contents of one's “second brain” be a work for hire (e.g., an interaction in which the end user owns the rights) rather than a software purchase. Thus, there will be an exponentially growing need for personal system administrators as new people enter the community of connected, collective, humanistic intelligence.

The End of Theft of Visual Attention

Linux has eliminated the need for pirated copies of poorly written commercial operating systems. Freely distributable software has resulted in improved operating system software and changed the nature of intellectual property.

Similarly, there is the issue of Humanistic Property. Humanistic Property was formerly free for others to steal, but now a technological means to prevent theft of Humanistic Property is proposed. This means that in the future, individuals will decide what advertisements they would like to see or not see.

For example, I am currently not interested in seeing advertisements for cars, cleaning products or condoms. However, I *am* currently in the market for certain components that I need to build the next embodiment of WearComp, so I would very much welcome the opportunity to see any advertisements by vendors of these products. I do not believe we will see the end of advertising, just the end of *unwanted* advertising—the end of theft of our visual attention.

Resources

Thanks to Kodak and Digital Equipment Corporation (DEC) for assistance with the Personal Imaging and Humanistic Intelligence projects.

Steve Mann inventor of WearComp (wearable computer) and WearCam (eyetap camera and reality mediator), is currently a faculty member at the University of Toronto, Department of Electrical and Computer Engineering. Dr. Mann has been working on his WearComp invention for more than 20 years, dating back to his high school days in the 1970s. He brought his inventions and ideas to the Massachusetts Institute of Technology in 1991, founding what was to later become the MIT Wearable Computing Project, and received his Ph.D. degree from MIT in 1997 in this new field he established. Anyone interested in joining or helping out with the “community of cyborgs” project or the RWM project may wish to contact the author by e-mail at mann@eecg.toronto.edu.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Polyglot Emacs 20.4

Jon Babcock

Issue #59, March 1999

A look at multilingual Emacs.

Ken'ichi Handa's Mule (multilingual Emacs) first appeared at the end of 1992. After almost five years, the Mule enhancements were included with GNU Emacs 20.x. For those of us who have yearned for multi-script capability since our first encounter with a computer (more than twenty years ago), it has been a long and often frustrating wait. We are now at GNU Emacs version 20.4 and things are finally beginning to look interesting to people who wish to work with multiple scripts on Linux. I am using Emacs for translation, exegesis and preparing reference material in multiple scripts.

Who wants to install a special Japanese Linux just to be able to read a Japanese source file for a translation job or read and write Japanese e-mail? I want to be able to include Chinese bibliographies, text and notes in the papers I write. I would also like to be able to include Tibetan or Greek scripts for philosophical or technical terms, along with their transliteration into Latin script. When I am discussing the structure of Chinese characters, I want to be able to make comparisons with Egyptian hieroglyphs. I want my quotations of French and German to look like French and German. I want to be able to publish all this on web pages as well as my PostScript printer. Some high-priced programs are coming out that address these issues, but Emacs 20.4 is here now. It runs on the best generally available operating system in the world, GNU/Linux, and it is free.

Figure 1

As an example of using Emacs to prepare multi-script reference material, let's look at the Buddhist numerical lists I am currently working on. Without much difficulty, I can write this list with a pen on a piece of paper. See Figure 1 for a bitmap of a page from my handwritten list. (Apologies for my poor calligraphy.) To get these scripts into a computer text file requires an input method specific

to each script. Fortunately, Emacs comes with Quail, which has a method to input each script in this example and many more.

To invoke Quail for Devanagari, use the Mule menu or type:

```
ctrl-x return
ctrl-\ devanagar-transliteration
```

Three other choices are available. For Tibetan, you will want **tibetan-wylie**. For Chinese, more than twenty methods are available. I use **chinese-py-b5** for traditional characters and **chinese-tonepy** for the simplified characters. The Quail Japanese input method is adequate for short strings, but not extensive input. This is one place where I feel a free input method editor for Linux is needed that equals or surpasses Microsoft's free Japanese IME for Windows. Wnn 4.2, the last free version of Wnn, worked well. I have used it in the past with Mule, but so far I have not be able to get it to work with Emacs.

Soon, I hope to add the Korean, Thai, Lao and Vietnamese equivalents of each term in each list. All these are supported by Emacs. Finally, I hope to add the Mongolian script, which is not yet available in Emacs.

As you start to use several different input methods, you may soon find the command to invoke them, **ctrl-x return ctrl-** cumbersome. I rebind it with:

```
meta-x global-set-key f3 return
set-input-method
```

While you're at it, you might as well bind the command **universal-coding-system-argument** to something handy—I use f2.

universal-coding-system-argument is the command that lets you specify which coding system you want Emacs to use when you execute your next command. If you do much multi-script work, this will probably be **ctrl-x ctrl-v return**, which revisits the file you just visited. On the revisit, Emacs uses the coding system you specified as the universal-coding-system-argument. From the main Emacs menu, you can select Mule/Set Coding System/Next Command to do the same thing. (See Emacs Manual 31.4.5 for details on rebinding keys.)

For information on each input method and sometimes a list of the characters you can use with it, type **ctrl-h I**. As usual in Emacs, tab will give you a list of choices if you don't know the exact name of the input method you are after. **ctrl-g** escapes from whatever you are doing in the mini-buffer. I said “sometimes” because some lists are missing. For example, in response to **ctrl-h I ipa**, Emacs returns “Input method: ipa (IPA in mode line) for IPA International Phonetic Alphabet for English, French, German and Italian” but provides no list of the actual symbols. For Devanagari, on the other hand, a full list of the letters

of the script is presented. Not given are the details on how to evoke several operations essential for being able to input the script properly. If you are familiar with the script, you can probably hack your way through. If, like me, you are a beginner and merely attempting to input it from a transcription in Latin script, even assuming your transcription is precise, you will not be pleased. Detailed descriptions of the various input methods are needed.

Start with Tibetan. Type the Wylie transliteration and the script appears—very smooth. For beginners, it is easier than writing Tibetan by hand. For the time being, I had to give up trying to input Devanagari. You may have better luck.

All these input methods come in a package called Leim. As of this writing, Leim is bundled with the Emacs-20.3.92 from ftp://ftp.etl.go.jp/pub/mule/.notready/, but it must be downloaded as a separate package for the Emacs pre-release from ftp://alpha.gnu.org/. Anyway, if Emacs cannot find all files included with Leim when it compiles on your system, you won't have any input methods. Let's hope the distributions will include Leim by default and give you an option to exclude it.

To evoke the multi-script capabilities of the new Emacs, another essential ingredient is Intlfonts 1.x. At present, it is version 1.2. This package provides all the fonts you need to display all the scripts. It, too, must be installed before you will find any joy in multi-script work.

The latest version of ps-print.el that comes with Emacs allows you to at least dump your multiple-script files to the laser printer. Currently, you must be content with “not scalable” bit-mapped fonts where one size fits all, but this is an essential first step. Perhaps CJK-TeX by Werner Lemberg (xlwy01@uxp1.hrz.uni-dortmund.de) or Omega, the new, purportedly internationalized TeX, will generously give us the ability to produce high-class, camera-ready, multi-script output for printing and a description of how to do it.

Figure 2

In Figure 2, we see the same text as in Figure 1 entered into an Emacs buffer, or as much of it as I could enter without a better understanding of the Devanagari input method.

It could be argued that we should not look to Emacs for help in printing, aside from the minimum requirement of being able to dump multi-script texts to the printer. Maybe the same holds for Web publishing—I don't know. However, it is frustrating to create a document in five or more scripts perfectly well in Emacs and then not be able to print it at a camera-ready level of quality, or publish it on the Web where the only character set that would allow the inclusion of all

five scripts, with some on-going support, is Unicode, particularly in its UTF-8 encoding.

If there were an option to map the multi-script file in Emacs to the Unicode character set and then save it in UTF-8 encoding, ithe file would be directly available as content for an XML or HTML document. True, some browsers cannot understand Unicode yet, and the browser user may not have all the fonts installed, but this is bound to change for the better soon. One thing for sure: few will be able to read it in Mule internal format or even in a mix of ISO-2022 character sets/encodings.

Regarding a Unicode converter for Emacs, Miyashita Hisashi took a first shot at a Mule internal-encoding-to-Unicode converter with his MULE-UCS converter, but as of this writing I have not been able to install it. I noticed on the Unicode mailing list that Mark Leisher (mleisher@crl.nmsu.edu) also has one under construction. Hopefully, by the time this article is printed, we will be able to produce a Unicode/UTF-8-encoded file from Emacs.

Emacs Help for the Translator

Once you have entered the multiple scripts into your Emacs buffer, you can marshal all the considerable forces of Emacs to work on your text. This is the big difference between Emacs and a program like Gasper Sinai's Yudit, for example. From a user's perspective, Yudit deals nicely with the encoding problems and it includes support for Unicode from the ground up, but most other features of a well-rounded editor have not yet been implemented.

Since Emacs is multilingual, it is also bilingual. Pick any two-language combination. I translate from Chinese to English (CE), mostly classical texts, and from Japanese to English, mostly commercial work for pay. Emacs has much to offer a JE or CE translator.

Emacs offers such helpful services as saving your cursor position in a buffer between sessions, saving the buffer arrangement of your Emacs session, splitting frames into Emacs windows and placing new frames in strategic locations on your big translator's virtual desktop, la FVWM2. It is also easily customized.

The five items of Emacs arcana that can be especially useful to a translator are saving and backup, outline minor mode, narrowing, abbrevs and bookmarks. See http://www.kanji.com/ for a more detailed version of the following.

Saving and Backup Using autosave

autosave is continuously taking care of the job of saving the latest version of your file. Use **ctrl-h v auto-save-interval** to check your current value for autosave. In the directory listing, an autosave version of your file is marked by pound signs (#), prefixed and suffixed to the filename.

I use **make-backup-file** to automatically produce sequentially numbered versions or drafts of my work. In this view, the file is complete right from the beginning, as soon as I save it with a name for the first time. Once my file has a name, it automatically becomes the “first version”.

It took a while to break my old habit of manually saving my file as often as possible, indeed each time I leaned back in my chair. Now I save only when I am ready to take a long break (like several hours or overnight), or when I feel I have reached the logical end of a certain draft version. The result is that I have only one autosave file at any given time, and a new draft (backup) file is produced at meaningful intervals. Use **ctrl-h v make-backup-files** to check or set. Use **ctrl-h v version-control** to check your current setting and the “customize” button found therein to turn on numbered backups.

By default, automatic deletion of backups will occur, thereby ruining the use of backups as a translator's simple version control system. Check the variables **kept-old-versions** and **kept-new-versions**. The default is 2, i.e., the first two and last two backups are kept; other backups are removed. To keep all backups, I set these variables to 500 each so a thousand backups will be kept before the ones in the middle are removed. After a translation job is finished, I usually delete them all.

Not only will you want to save your files, but also your constellation of visited buffers. This is done through the use of the Emacs desktop. Add these three lines to your .emacs file:

```
(load "desktop")
(desktop-load-default)
(desktop-read)
```

You must use **meta-x desktop-save** to initiate this process and then start Emacs from the same current directory each time you need to recover this state.

You can also save state within an Emacs session by using a register.

Outline Minor Mode

Until a full-scale major mode for translation is written, **outline-mode** or **outline-minor-mode** must take on the responsibility of managing the source text, the target text and related notes, comments and references.

In outline mode, there are two kinds of lines: header lines and body lines. Header lines start with a star in the leftmost column. The more stars, the deeper the level into the outline. One star means the line is at the top level. For short jobs, I have only one top-level heading line with the words TOP LEVEL or the title of the job. In longer jobs, I use it for Part One, Part Two, etc. I put contact information that applies to the whole job and notes about the deadline, size, charges and any special provisions as body lines below this top-level heading.

Within my translation environment, body text means notes and commentary. I find it extremely convenient to embed these directly in the working file rather than keep them as separate files. Such inter-linear notes are thus permanently welded to the text to which they relate.

Outline level 2 is always the Japanese or Chinese source text and level 3 is always my target text, the English translation. If I suspect there is a typo in the Japanese source text, my proposed correction can appear in the body lines connected to those level 2 heading lines, i.e., to those particular lines of the Japanese source text. Likewise, definitions, questions, reference sources, comments and notes on words, URLs and anything else that throws light on the translation are set down in body lines that are directly connected to the level 3 heading lines, which are always the English target text I am writing.

When I am finished translating, a keyboard macro strips out the level 3 lines (my translation) and produces the file that will go either directly to the client or through some unavoidable conversion, and even formatting, in a word processor running on Linux, such as ApplixWords.

It would be nice to see **outline-mode** generalized into a “show and hide” mode so that you could show body lines alone at whatever level you choose.

Narrowing

In effect, **outline-mode** or **outline-minor-mode** gives us a pre-structured kind of narrowing. Narrowing is more general in the sense that it can be arbitrarily applied to any portion of text in the buffer. Furthermore, with **outline-mode** or **outline-minor-mode**, it is quite possible to edit large chunks of the buffer that are not currently displayed. For example, if you delete or move the heading line, the entire entry under it including its body lines is deleted or moved. Narrowing, on the other hand, restricts editing to the narrowed portion alone. Place a mark at one end and point at the other. Then type **ctrl-x n n** and the accessible portion of the buffer will be reduced to precisely that region only. See the right-hand buffer in Figures 3 and 4 for examples. In both, the right-

hand buffer is narrowed. Only the accessible portion is available for editing. **ctrl-x n w** widens, to make the entire buffer accessible again.

Figure 3

Figure 4

In Figure 3, what you see is reduced to just the Japanese source text and the English target text, i.e., level 2 and level 3 outline "heading lines". In Figure 4, what you see is expanded to include snips from on-line dictionaries, notes and comments, i.e., outline "body lines". But the accessible portion produced by narrowing is the same in both cases. For Figures 3 and 4, I used outline mode to keep source text, translation, abbrevs, glossary entries, notes and commentary all in one file. The left buffer shows only level 2 outline header lines, i.e., the Japanese source text, whereas the right buffer in Figure 3 shows this plus the target text and Figure 4 shows all three: source, target and notes.

Abbrevs

I use abbrevs primarily to help enforce consistency on my English target texts but also to avoid some typing.

Let me take an example from my non-commercial work but which applies to all types of CE/JE translation as well. Buddhism has a large vocabulary of "technical terms" that constantly reappear. In the Buddhist texts I work on, five of the most frequent are:

- prajnaparamita **->** pp
- mahaprajnaparamita-sutra **->** mpps
- utmost, right and perfect enlightenment **->** urpe
- bodhisattva **->** bs
- the buddha said **->** tbs

With **abbrev mode** turned on (**meta-x abbrev-mode**), typing **bs** followed by a space instantly inserts **bodhisattva**, **pp** inserts **prajnaparamita**, etc. With **abbrev mode** turned off, I can still force an insert before point (the position of the cursor) with **ctrl-x a e** (**expand-abbrev**).

When I am working on a commercial, technical JE translation job related to Linux, for example, I want to forget about Buddhist-related abbrevs, so I save and load files of abbrevs as appropriate.

Bookmarks and Registers

Bookmarks have arbitrary (long) names and remain from one Emacs session to another. Registers have one-letter names and disappear as soon as a session is ended. A bookmark tells Emacs where to go in a buffer, whereas a register stores data to insert into a buffer, such as a filename, a window configuration, a piece of text or a rectangle. Yes, a register can also store a location and thereby pretend it is a temporary bookmark.

You would think the proper function of a bookmark is simply to mark your place in a file so that whenever you revisit it, you will go to that location; however, I don't need bookmarks for this. I usually leave off at the location I want to return to, so that simply switching buffers via command, menu or **ctrl-mouse_button_1** takes me back to where I was. Revisiting a file takes me there too, because I have **place-saving** turned on. (Use **toggle-save-place** to turn it on for a specific file and **setq-default save-place t** to turn it on globally for all files. [Added in Emacs version 19.19.]) Therefore, I feel free to use bookmarks for something less ordinary. **ctrl-x r m *bookmark_name*** sets the bookmark; the bookmark name is a string of Japanese or Chinese text, i.e., my glossary entry.

In Japanese, Emacs automatically sorts the kana bookmark names. (Kana are the graphic forms used to write the Japanese syllabary.) When I click on a bookmark, I am propelled directly to the work file where that word or phrase resides in an actual job. Beginning with Emacs 19.29, bookmarks could carry annotations. (Type **ctrl-h m** after **meta-x list-bookmarks** for more information on annotations.) I have co-opted this feature to carry the English glossary or meanings of the glossary entries (i.e., the bookmark names). If the annotation is sufficient, I don't need to visit the glossary entry in its location within a working file. If I want to see the context in which I have translated the word, phrase or sentence, one click lets me take a look, regardless of whether that file is currently being visited by an Emacs buffer or not. I can display the location in the file in a buffer that replaces my current bookmark editing buffer or have Emacs put it in a separate window adjacent to the buffer holding the bookmark list. I can also just ask Emacs to tell me the name of the file where the bookmark can be found without actually displaying it.

The one drawback to this neat, but nonetheless makeshift device, is that I can assign only one location to each glossary entry. If I want to add more, I must put them in the annotation.

Help from Other Programs

Finally, for JE work, Emacs is not alone. As planets circle the sun, I surround Emacs with several programs that help me figure out the Japanese or Chinese

source text or that aid in delivering it in a form demanded by a client. My assistants include the following:

- **Sumomo**: a parser of Japanese syntax (no mean feat in a script where there are no spaces between words)
- **Kakasi**: a converter of kanji into kana or Romaji (see Figure 5)

Figure 5

- **Xjdic**: Jim Breen's JE Dictionary which, beside being extremely useful as a dictionary, is a splendid example of a collaborative, Net-based project (see Figure 6)

Figure 6

- **Bookview**: a front end to **ndtp**, a server of Japanese electronic reference books (see Figure 7)

Figure 7

Also available are many nifty GNU or GPL-licensed programs that circumambulate the Linux kernel and can be called into service at a moment's notice, such as **ls**, **wc**, **grep**, **xdiff**, **telnet**, **ftp**, **wget** and **fvwm2**. Finally, there is Mozilla, Star and Corel Office, ApplixWare, the American Heritage Dictionary and the Oxford English Dictionary.

Whatever your arrangement, to be productive it must be consistent and persistent, i.e., for a certain task (like CE or JE translation) you must be able to recreate the same arrangement quickly. I have an init.hook file configured so that these applications all start up when I start fvwm2. By renaming it, I can immediately set up a different desktop for a different task. (See Figure 8 for one possible layout of a translator's desktop.)

Figure 8

With Emacs 20.4, we have the first version of GNU Emacs that is equipped to function as the main tool in a multi-script text worker's workshop.

Resources

Jon Babcock lives on the old family homestead in Montana, far from the maddening crowd, studies karma and dharma and kanji culture and, when necessary, translates Japanese for pay. He started using GNU programs and Linux in 1992. He can be reached via e-mail at jon@lotus.kanji.com.

Smart Cards and Biometrics: Your Key to PKI

David Corcoran

David Sims

Bob Hillhouse

Issue #59, March 1999

The cool way to make secure transactions.

For centuries, security was synonymous with secrecy. The shared secret—conducting business between two parties who each knew the code—was a worldwide approach. Even in this age of electronics and supercomputers, passwords and PINs are shared between you and the computer or ATM machine to which you want access. But secret passwords require a great deal of trust between parties sharing the secret. Can you always trust the administrator or other users of the machine you are accessing? Most computer break-ins today are due to compromise by system users or by a hacker who uses a legitimate account (possibly yours) to gain access to general security—sometimes even gaining superuser access. This traditional paradigm of shared-secret computer security could soon be a thing of the past with smart-card-based cryptographic credentials and biometric authentication for access control.

Some individuals and companies are replacing shared secret security (also called symmetric security) with the Public Key Infrastructure (PKI) approach. PKI uses a standardized set of transactions using asymmetric public key cryptography, a more secure and potentially much more functional mechanism for access to digital resources. The same system could also be used for securing physical access to controlled environments, such as your home or office.

In a PKI world, everyone would be issued at least one cryptographic key pair. Each key pair would consist of a secret (private) cryptographic key and a public cryptographic key. These keys are typically a 1024-bit or 2048-bit string of

binary digits with a unique property: when one is used with an encoding algorithm to encrypt data, the other can be used with the same algorithm to decrypt the data. The encoding key cannot be used for decoding. Public keys are certified by a responsible party such as a notary public, passport office, government agency or trusted third party. The public key is widely distributed, often through a directory or database that can be searched by the public. But the private key remains a tightly guarded secret by the owner. Between sender and receiver, secure messaging (or other secure transaction) would work as described below.

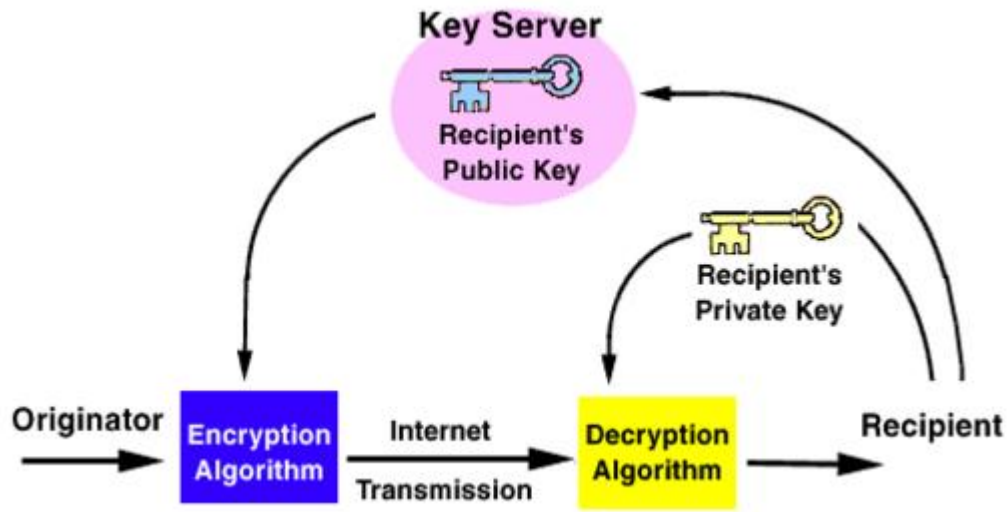


Figure 1. Sender

For the sender (Figure 1), the following steps occur:

1. Message data is hashed; that is, a variable-length input string is converted to a fixed-length output string. Hash functions are mainly used with public key algorithms to create digital signatures.
2. A symmetric key is created and used to encrypt the entire message. DES and IDEA are examples of symmetric key cryptography.
3. The symmetric key is encrypted with the receiver's asymmetric public key.
4. The message hash is encrypted with the sender's asymmetric private key, creating a digital signature independent of the encrypted message.
5. The encrypted message, encrypted symmetric key and signed message hash are sent to the receiver.

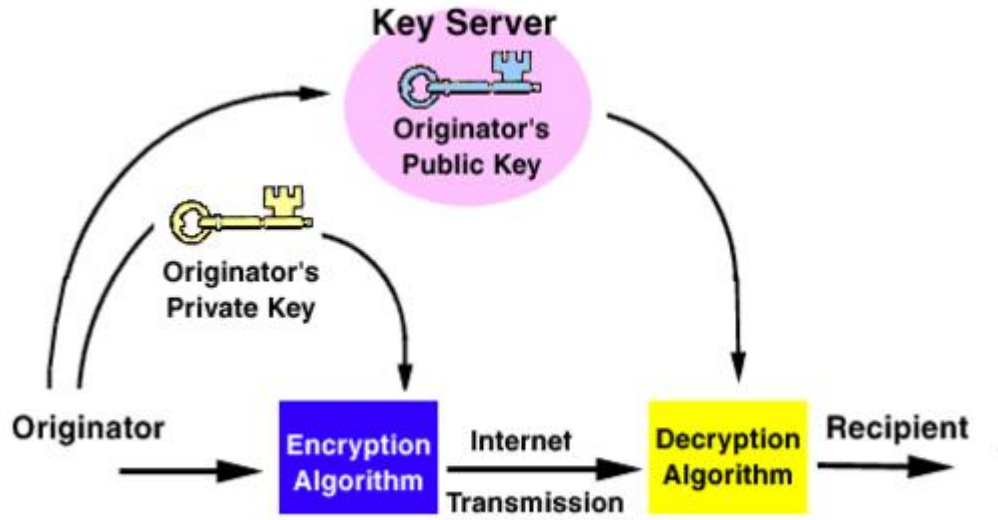


Figure 2. Receiver

For the receiver (Figure 2), these steps occur:

1. The encrypted symmetric key is decrypted using the receiver's asymmetric private key.
2. The symmetric key is then used to decrypt the message body.
3. The encrypted hash is decrypted with the sender's asymmetric public key.
4. The decrypted message is then rehashed with the original hashing algorithm.
5. The two hashes are compared to verify the sender's identity and serves as proof that the message was not altered in transit.

Several technical issues must be solved before this scenario can be realized:

- Secure key storage
- Secure authentication to the key store
- Directory services (central public key database)
- Key escrow or other emergency recovery method for encrypted data
- Cross-platform standards (Microsoft PC/SC, Netscape SMIME, Intel CDSA, IBM OSF, etc.)
- International export and usage regulations for strong cryptography

If these issues seem daunting, remember how impossible a common network strategy once seemed. Today, the Internet is everywhere and connected to almost every type of computer. Let's consider the PKI issues one by one.

Secure Key Storage

A few possibilities exist for key storage. Remember that these keys are long bit strings and memorizing them is out of the question. There are three storage possibilities: hard disk, floppy disk and smart card.

Hard disk storage provides a low-cost solution. The user's key pair is stored directly on the user's machine. The user authenticates with a shared-secret password to unlock their private key for signing. This solution does not allow for easy mobility in a desktop environment, and it put constraints on terminal applications. A user cannot easily use another machine without offloading the information to the machine they are trying to use. Problems also arise if a user tries to log in to another machine and the network is unavailable. Hard disk storage also lends itself to physical attacks, such as theft of the terminal or hard drive, both of which can be used for leisurely password cracking.

Floppy disk storage addresses the mobility problem. Now the user has a quite inexpensive and portable way of using mobile digital credentials. Someone could use another machine simply by inserting the floppy and using a traditional shared secret password for authentication. However, floppy disks are not well known for long term, robust data retention, particularly when carried around. Floppy disks are susceptible to magnetic fields (airport security stations, for example) and do not fit conveniently into your pocket or wallet.

Smart card storage presents the best scenario. Smart cards have been used for many years in Europe for a variety of applications. These credit-card-sized computers have a rugged and familiar form that fits nicely into a wallet or pocket and can take lots of physical stress. Some modern processor-based smart cards even have on-board cryptographic co-processors that allow signing and key generation to be done entirely on the card, so the private key might never need to be revealed or offloaded. The microprocessor gives the smart card a big advantage over magnetic or optical media storage. A smart-card-based PKI might be very secure indeed, eliminating any possibility of the key pair being snooped out during creation and transmittal. The initial expense is higher, since smart cards require a reader. But this additional cost is offset by much higher security for the private key and by convenient porting.

Secure Authentication with Biometrics

All three storage methods—hard disk, floppy or smart card—could use symmetric encryption (shared-secret password or PIN) to secure the private key. This accommodates single sign-on, since once a user authenticates to the key store, cryptographic protocols can be used for subsequent authentication to different applications. This would be good from an administrative point of view, but makes the security situation much worse. Now, if a cracker gains

access to the shared secret password or PIN that secures the cryptographic keys, he also gains access to every cryptographically protected application or data element available to the authorized user. What we need is an authentication method to which only the authorized party has access. Enter biometrics.

The word biometrics comes from the Greek words bio and metric, meaning “life measurement”. By measuring something unique about an individual and using that to identify them, we can achieve a dramatic improvement in security of the key store. Newer biometric measurements include DNA from tissue samples, voice pattern, face pattern or even the arrangement of blood vessels in the retina or pattern of coloration in the cornea of the eye. The oldest and most widely accepted biometric is the fingerprint. The tip of every finger has a characteristic called “friction ridges”. While generally similar, no two friction ridges are exactly the same. By imaging the ridges of the fingertips, we get the fingerprint.

Most implementations of fingerprint biometrics create a template from the original image, which is a fraction of the size of the original fingerprint image. This template can be used only to compare the fingerprint against other templates, and it cannot be used to recreate the original image. Template implementations of biometrics fit well with smart cards for two reasons. First, they usually range from 100 to 600 bytes in size and can easily fit on a smart card. Second, you don't have to worry about an attacker reproducing your fingerprints from your templates and using them to impersonate you. Biometrics can aid authentication. Here is a rough outline of the procedure for authenticating yourself to a computer application:

1. Insert your smart card into a reader. The smart card contains your cryptographic keys and biometric fingerprint data.
2. Enter your shared-secret PIN (or password), in order to unlock the digital representation of your fingerprint. In the trade, this is known as the minutia data.
3. Place your finger on the scanner. The scanned fingerprint is compared to the fingerprint data on the smart card.
4. If the data matches, the smart-card fingerprint data is converted into a number and combined with the smart-card secret PIN (retrieved in Step 2) and used as a symmetric cryptographic key to decrypt the private key.
5. A nonce (random number) is passed from the computer application to the smart card.
6. The private key on the smart card is used to encrypt the nonce and pass it back to the application.

7. The application verifies that a certified public key obtained from the network-based directory service or from the card does, in fact, decrypt the encrypted message from the card and reveal the same nonce that was originally passed to the card.

This process irrefutably authenticates the person presenting the card as the same person to whom the cryptographic keys belong and provides the necessary tight binding between the cryptographic key storage and the authorized user of the cryptographic keys.

By this time, you are probably asking, “Just how well do these biometrics work? What is the margin for error?” Two terms describe the functionality of biometrics. The false acceptance rate (FAR) is the probability that an intruder is accepted with a measurement that does not belong to the enrolled user. The false rejection rate (FRR) is the probability that an enrolled user is not recognized. Good biometrics have low FAR and low FRR, but unfortunately few standardized tests are available to determine these results, as each biometric read takes measurements in a slightly different way. Thus, third-party verification is very important when evaluating any biometric.

As a rule, there is usually a trade-off between security and convenience. Biometrics are no exception. In general, the better their security (the lower the FAR), the more inconvenience there is to the user, because more false rejections occur. Similarly, the more convenient the system is to use, the poorer the security is. Good biometric systems allow the user to choose from a wide range of possible FAR/FRR levels, so that convenience can be maximized for the desired level of security.

Exposure to a few detective movies can cause the average person to ask the macabre but pertinent question, “How can these devices ensure that the finger is alive?” Attempts have been made to solve this problem. Some vendors measure the heat of the finger to ensure that it is at body temperature, which makes the system difficult to use in cold climates or with people who are predisposed to cold hands. Other vendors measure the conductivity of the finger to prevent forged fingerprints (for example, silicone castings). This does not address the dead finger issue, but it is worth noting that conductivity measurements can be fooled with a bit of saline solution on the silicone casting. The best solutions spectroscopically measure the amount of oxygenated hemoglobin in the blood, as this measurement is the most difficult to fool and is radically different for live and dead fingers, but again there is a trade-off between price and necessity. Are you authenticating yourself to a space shuttle launch or the garage door?

Figure 3.

One of the best products to merge smart-card technology and biometrics is the BioMouse Plus from American Biometric Company (http://www.biomouse.com/). The BioMouse Plus is an integrated smart-card reader and fingerprint scanner. A Linux toolkit is provided for developers, with documentation on how to create biometric and smart-card aware applications. The toolkit is complete with examples, sample source code, drivers and libraries. In fact, over 13 platforms are supported, including most flavors of UNIX, Windows and MS-DOS.

The Central Public Key Database

Directory services play an essential role in any PKI system. Applications must be able to verify the certificate authority of the public key contained on the smart card. The certificate authority is the organization that initially issued the encryption keys and smart card. The certificate authority verifies that the person is who they claim to be. If privacy concerns can be overcome, public keys (for the certificate authority and for the individual) should be available to all applications that need cross verification.

PKI at the office: a person has a smart card containing cryptographic keys secured with biometrics and signed (validated) by a government agency. Now the person applies for a job in the private sector. If the company verifies that the government signature is valid, the person's public key can be used for employment verification. The smart card is essentially reusable as identification.

Personal Banking: this application makes a binding between the application, the public cryptographic key and personal data stored in an employee directory. Again, the original single identity token is reused. Directory services and biometrically secured cryptographic key storage would truly enable electronic commerce. Such a scheme, if widely adopted, would allow an individual to carry a single convenient token to authenticate themselves to applications anywhere.

A Variety of Standards

Assuming that a smart-card-enabled PKI works for all other reasons, a few issues must still be overcome with regard to standards and cross-platform performance. The smart-card environment needs standard resource managers

and APIs for communicating to the card via the card reader. These APIs are generally card-specific. Some APIs are reader-specific. Since most smart cards adhere poorly to common standards such as ISO-7816-4, it is necessary to have a high-level API for communicating to all cards. The same is true for readers. Generally, a reader's resource manager tracks the different readers installed on the system and monitors events such as card insertion and removal. This resource manager is also responsible for transferring control of the smart card to other applications, so that multiple applications can communicate with the card.

Card management tracks communication speeds and the currently selected file. Consider the following example: application B wants to offload data from elementary file 0200. Application A is in wait state but currently has selected file 0001. The card manager must keep track of this file so that when application B takes control, selects 0200 and performs data transfer, application A can regain control upon completion and reselect elementary file 0001. Without such resource management, a user must assume another transaction has occurred and do a cold reset before any file or verification-related transaction.

Cards contain specific functions that make them unique. The most common is the cryptographic-capable smart card. In this card, it is necessary to have yet another common API which communicates with the card manager. This API is known as the cryptographic service provider. The cryptographic API performs functions such as key generation, secure signing, hashing, encryption and key verification.

Just as several standards exist for card and reader resource managers, quite a few proposed standards have been made for cryptographic service providers. One of these is the PKCS-11 standard, driven mainly by Netscape. Microsoft, of course, proposed a different standard, called the Crypto API (CAPI). Intel is also making a run at the cryptographic middleware market with the release of CDSA. CDSA is more of a framework than an API and takes advantage of CAPI and PKCS-11. CDSA and PKCS-11 both lack one major component for a system: card and reader management. Neither CDSA nor PKCS-11 was designed specifically for cryptographic tokens, but both would fit nicely with other card and reader managers. Microsoft's model encompasses a specification known as PC/SC. PC/SC handles all card and reader resource managing and fits with the CAPI for cryptographic support. All of these specifications can be found at http://www.smartcardsys.com/.

On the open standards end, IBM has created support for reader and card resource managing in a cross-platform style using Open Card Framework (OCF). This is a purely Java-based card and reader resource manager that runs on most operating systems with a working JVM (Java Virtual Machine), including

Linux. Nice idea, but what is missing? OCF fails to include cryptographic support, although an open version of PKCS-11 would probably fit nicely on top of the infrastructure. If this PKCS-11 is written in ANSI C, then users of superior workstations such as Linux, Macintosh and Sun could have all the support included on Microsoft systems. A port of CDSA for non-Microsoft operating systems would also be nice, since one could imagine better portability to a Microsoft OS. In fact, a PC/SC-compliant resource manager for non-Microsoft systems would limit cross-platform compatibility only by the low-level reader driver code.

The MUSCLE project is currently working on a C-based resource manager for smart-card readers. The resource manager uses remote procedure calling to make remote authentication possible. For more information, visit http://www.linuxnet.com/smartcard/.

Export Regulations

The laws regarding export of strong cryptography are a patchwork quilt at best; collectively they represent possibly the largest hurdle to be overcome. Solutions that employ message recovery features such as multiple key encryption or key recovery will help move legislation forward. Currently, the worldwide nature of the Linux development community and the modular approach of the MUSCLE project would seem to facilitate the spread of this technology.

Conclusion

Integrating smart cards, biometrics and public key cryptography provides a solid foundation for developing secure applications and communications. The highest level of security uses three-factor authentication:

- Something you know (password or PIN)
- Something you have (smart card, magnetic stripe card or a physical key)
- Something you are (your fingerprint, retinal/iris scan or voice pattern)

An individual gains three-factor authentication by combining a smart card, biometric and PIN. If the user loses the smart card, the card is inoperable without the biometric. Forged fingerprints are weeded out with use of the PIN.

In a smart-card-secure world, you are not locked into one form of authentication, such as the ever-vulnerable password. You control your identity because it is contained on the card you carry with you. Even if attackers run Crack 5.0 on your Internet provider's password file, they cannot gain access without possession of the smart card tucked safely in your own front pocket.

The argument for improved security is a noble one. Some methods of achieving improved security may use expensive hardware and still be relatively easy to compromise. Most symmetric forms of security fall into this category. It is only a matter of time before a shared secret is no secret at all. Smart cards combined with biometrics provide today's best approach to secure electronic data. But as your mother may have told you, the only way to truly keep a secret is never to share it.

David Corcoran is a student studying Computer Science at Purdue University. He works with the COAST/CERIAS labs directed by Gene Spafford and also as a Knowledge Worker for Schlumberger Limited in Sugar Land, Texas. He can be reached at corcordt@cs.purdue.edu.

David Sims is the Technical Manager of Information Technology for Schlumberger Limited, based in Sugar Land, Texas. He holds a BS degree in Mechanical Engineering from Washington University in St. Louis, Missouri. He can be reached at sims@sugar-land.sl.slb.com.

Bob Hillhouse is a Senior Software Engineer for American Biometric Company. Based in Ottawa, Ontario, Canada, he holds a BMath degree in Computer Science with Electrical Engineering Electives from the University of Waterloo. He can be reached at bob@abio.com.

Archive Index Issue Table of Contents

Advanced search

Linux for the International Space Station Program

Guillermo Ortega

Issue #59, March 1999

An overview of two applications for spacecraft and why these applications are being run on Linux.

The first element of the International Space Station (ISS) has already been launched from Baikonur, Russia. ISS is the biggest civilian endeavor ever entrusted to human science and technology. Thousands of software code lines are being, and will be, written for the station, on-ground and on-board. ISS represents a synergy among many space agencies and companies around the globe, including European Space Agency (ESA). ESA has been developing several prototype systems which will pave the technology road for the European contributions to the ISS program. Among others, ESA has chosen Linux as the operating system for two software products which will control the rendezvous and docking operations for a servicing spacecraft called ATV. This article presents an overview of these products, explaining why they run on Linux, the advantages and disadvantages of doing so and the future of Linux in the space industry.

The Automatic Transfer Vehicle

The ESA Automatic Transfer Vehicle (ATV) is an unmanned spacecraft planned to perform regular re-boosting and re-fueling of the International Space Station. Other ATV missions will comprise payload supply and payload removal from the ISS. The ATV is an ongoing project approved in October 1995 by the Council of the European Space Agency. ATV is scheduled to be launched for the first time by an Ariane 5 rocket from Kourou, French Guyana in February 2003.

Figure 1. ATV Spacecraft

In its early configuration, ATV was designed as a cylindrical shaped spacecraft containing a cargo module (pressurized or unpressurized), a docking port and a

propulsion module. Lately, ATV was modified with four solar panels added (see Figure 1).

The ATV mission profile establishes docking with the ISS in the Russian segment of the station. Rendezvous operations start at about 20km behind the station. This means that ATV will fly behind and faster than ISS, in order to catch up to the docking port of the station. The problem of the space rendezvous is the mating of two spacecraft in orbit—a small active chaser spacecraft (ATV) and a big target (ISS).

Although spacecraft rendezvous and docking may look simple, they are not. The mathematical equations governing the relative movement between chaser and target are rather complex. The onboard control of the rendezvous operations is to a large extent automatic, but not fully autonomous. Some control tasks are done from the ground control center and others onboard the station. Although the ATV onboard computer is fault tolerant, the complexity of the mission does not allow the prevention of and recovery all possible types of mission failures. Special features of the onboard system would allow detecting and forecasting failures, isolating them and proposing and executing recovery actions, but only for those types of contingencies anticipated during system design. The failure detection and recovery features are linked to both mission safety and mission success.

Linux to the Rescue

To overcome these and other issues, ESA built two products: GOAS and RACSI. GOAS is the Ground Operator Assistant System for the rendezvous operations of ATV. Used on ground, GOAS is a software tool to monitor the ATV mission and intervene in case of a problem. GOAS provides complex command and control capabilities to replan the entire mission if necessary. RACSI is the remote ATV control at ISS. RACSI is a laptop computer running a software package operated by an astronaut onboard the station. RACSI double monitors and checks the ATV mission and provides two simple command capabilities: temporarily interrupt the mission or command a collision-avoidance maneuver.

Currently, both the GOAS and RACSI developments run under Linux. Although GOAS was developed on Solaris (using versions 1 and 2), the software was ported to Linux without difficulties. RACSI was originally programmed entirely under Linux. For both systems, Linux was chosen as the underlying operating system because it provides four basic features required by up-to-date space applications: reliability, performance, portability and affordability. Reliability is crucial to space applications. The feature of reliability is guaranteed by the robustness of Linux: both applications run dozens of processes concurrently, using extensively shared memories and semaphores. The software never

crashes or misbehaves, despite the fact that both systems were designed to run nonstop for weeks or even months.

Performance is the definitive factor in measuring real-time critical software. Although Linux is not used in real-time mode (the RT-Linux module is not loaded), the applications run in real time. That is, they receive data from the spacecraft, display it and send it back to the satellite, all in real time. Everything runs within the specified communications rate between craft.

Software portability is of vital importance for upgrades and applications enhancement. Portability among UNIX flavors can be done quickly, preserving expandability and keeping manpower costs down. This is not true for other non-UNIX operating systems. In addition, Linux is available for an enormous range of hardware platforms, making the change between platforms as simple as recompiling (in most cases).

Nowadays, space applications often lack the funds needed to buy costly licenses. Linux is a zero-cost operating system, which provides true affordability. It can be copied as many times as desired, keeping license costs and royalties low. This is true not only for the operating system but also for the tools (compilers, debuggers, editors, development environments, etc.) which come with it.

RACSI in Space

Figure 2. RASCI Screenshot

RACSI runs on an IBM ThinkPad laptop. The software requires 64MB of RAM and occupies around 40MB of disk space. (See Figure 2.) This type of laptop was chosen due to the hard radiation resistance requirements on the space station. It also provides an XGA graphical screen with a resolution of 1024x756 and 64K colors, mandatory for displaying trajectories and spacecraft equipment status. RACSI uses the X Window System (X11R6) with FVWM as its window manager. The Linux distribution currently installed on the laptop is Slackware version 3.0 with kernel 2.0.30.

A desktop configuration system has been installed on a Pentium Pro, running S.u.S.E. 5.2 with kernel 2.0.33. The pointing device of the system is the laptop track ball, although a conventional mouse can also be used.

RACSI was programmed entirely in ANSI C. It uses the Moo-Tiff libraries from InfoMagic version 2.0, so all widgets are Motif type. (FVWM fulfills the complete look appeal of the application.) The system uses shared memories and the Linux file system as database storage for telemetry data coming from the spacecraft. RACSI process interfaces make use of application program

interfaces (APIs). The APIs constitute an additional layer of software, which allows hiding the differences between local and remote software calls between applications. The software of RACSI is subdivided into several modules (see Figure 3) according to a modular architecture compliant with ESA software standards.

Figure 3. RACSI Architecture

Telemetry Handler: this function is a data-handling module. It receives all incoming types of data from the spacecraft and preprocesses them to generate an additional set of data valid for storage and archiving. Next, this module distributes the data to a predefined set of client applications (e.g., Mission and Vehicle Monitoring and Control).

Mission and Vehicle Monitoring: this module allows the astronaut to have an overview of the status of both the mission and the vehicle. The Mission and Vehicle Monitoring module extracts data from the Telemetry Handler function and sends it to the Information Presentation function. This function provides a set of displays showing, with different levels of detail, information related to the mission and the vehicle extracted from the telemetry or resulting from any other data processing.

Failure Detection and Assessment: this function performs the detection and identification of a failure. When a failure is detected, the astronaut is free to manually interrupt the mission (stop the spacecraft) or abort the current mission plan (give control to ground).

Display Management: the purpose of this function is to provide an on-screen data presentation to the astronaut. RACSI screens are organized in three areas (see Figure 2): mission display (with information related to mission phase, mission transitions, etc.), main display area (trajectory plotting, equipment surveillance, etc.) and messages display area (local messages and warnings).

Telecommand Handler: this module provides centralized services to send two possible orders to the spacecraft: mission interrupt or a collision-avoidance maneuver.

GOAS on the Ground

Figure 4. GOAS Screenshot

The native GOAS system runs on a Sun workstation, Ultra-SPARC 5, with 64MB of RAM and 300MB of hard disk space. A color monitor is mandatory. (See Figure 4.) This configuration runs under Solaris (version 2.5). It uses the X (X11R6) graphical user interface, with Sun OpenLook as the window manager.

The Linux version was developed from this initial system; the Linux GOAS runs on a 233MHz Pentium with 48MB of RAM. This system also uses X11R6, with Linux OpenLook as the window manager. The pointing device is a mouse, although using keystrokes is allowed.

GOAS was programmed in the C and C++ languages. C is used for programming the applications, C++ for programming the graphical interface. GOAS uses a collection of commercial off-the-shelf software routines (ILOG views) to build certain parts of the man-machine interface.

GOAS can run with two or three monitors at the same time, allowing the viewing of many spacecraft parameters. It is possible to configure a different monitor to display the man-machine interface of the monitoring, failure detection, replanning modules, etc. Like RACSI, the software of GOAS is subdivided into several modules (see Figure 5), according to a modular architecture compliant with ESA software standards. However, GOAS is much more complex in terms of mission planning capabilities, trajectory prediction and failure detection and recovery.

Figure 5. GOAS Architecture

The Telemetry Handler receives the data from the spacecraft, archives it in several databases and broadcasts it to the necessary clients in the system.

The Failure Assessment subsystem performs the detection and identification of a failure by running a set of automatic tests under the control of the ground operator, are implemented at the mission, guidance, navigation and control levels, and compare both the actual and predicted state with reference corridors for position, attitude, rates, etc. This module archives data in a database.

In case of problems, the Failure Assessment module decides which recovery procedure should be applied to recover the mission: the Fast Intervention module when the recovery must be immediate, the Short-Term Recovery actions or a complete Mission Replanning.

Fast Intervention: the emergency actions managed by this function allow interrupting the mission. The ground operator is confronted with a predefined set of emergency actions: stop the spacecraft, initiate a drift out of the station inhibiting the thrusters, or initiate a collision-avoidance maneuver directly controlling the spacecraft thrusters. This strategy will prevent collision with the station.

Short Term Recovery: this function is used in situations where recovery allows a period of time between 5 and 15 minutes. The goal is to stabilize the mission, and afterwards to start the Replanning function once that situation is reached. If safety is not compromised, different Short Term Recovery actions can be launched: force switching of some equipment to the redundant ones, quick study of the impact of the coming maneuvers, activation of a short sequence of maneuvers, etc.

Mission Replanning: the purpose of this function is to support the operator intervention when the mission needs to be changed. It is a feature not present in RACSI—an astronaut cannot replan a mission; only the ground control center can. The Replanning function is launched on request. It supports the ground operator in the definition of a new mission plan according to three items: replanning scenario, mission constraints and status of the chaser equipment. This function can be run in automatic mode (the computer replans a whole new mission without operator intervention), semi-automatic (the operator is asked for some parameters) or manual (the operator constructs the maneuver sequence step-by-step). Visual information is constantly displayed to the operator.

Conclusions

The success of Linux is grounded in the fact that the work created by one group of people is not owned by any other group of people. Reliability, performance, portability and affordability are the four characteristics which convinced ESA to use it for real-time spacecraft control software. Important work still needs to be done; hopefully, the coming kernels will be POSIX compliant, plug-and-play will be truly available and multimedia capabilities will be extended beyond user expectations. I am almost certain that Linux will run onboard the International Space Station or in any of the ISS components' ground control centers around the globe. Linux has earned its excellent reputation and can successfully compete with all other available operating systems.

Guillermo Ortega works in the guidance and navigation area of the European Space Research and Technology Centre in the Netherlands. He has been working with Linux in space projects since 1994. He can be reached via e-mail at gortega@estec.esa.nl.

Archive Index Issue Table of Contents

Advanced search

Chris Brown of Learning Tree International

Marjorie Richardson

Issue #59, March 1999

Linux enters the mainstream as companies such as Learning Tree and Caldera offer training courses for Linux. Here Learning Tree tells us why they are doing it.

Adding further weight to the growing acceptance of Linux as an industrial-strength operating system, Learning Tree International, a world-wide supplier of technology training, recently announced a four-day, hands-on course aimed specifically at users and administrators planning to deploy Linux in a commercial environment. We at *Linux Journal* feel strongly that training courses for Linux will help bring it into the mainstream marketplace. With this in mind, I talked to Chris Brown, Product Manager for UNIX and C/C++ Courses, by e-mail during October.

Marjorie: Tell us something about your UNIX courses and why you decided to begin offering Linux courses.

Chris: Learning Tree has offered a wide variety of UNIX courses for several years, and until recently, most of them used Solaris as a platform for the in-class exercises—some of them still do. However, an increasing number of courses ranging from UNIX Programming to UNIX Security are now taught using Linux.

The company also uses Linux in courses outside the UNIX curriculum, including some of the TCP/IP, networking and web courses.

We had to consider the dilemma of the free software culture versus the non-free Learning Tree training culture. Would people who used free software be willing to pay for their training? Only time will tell. Certainly, if you look at the total cost of ownership, paying for Linux training makes just as much sense as paying for NT training.

John Moriarty, who heads up our product development team, had these comments:

> We were rather nervous at first about introducing Linux into the classroom, not on technical grounds, but simply because we wondered how our customers would react. Having paid good money for their training, would they mind finding a free operating system on the machines in front of them? So far, I haven't heard a single complaint. In fact, some attendees get really fired up to discover that for very little money, they can practice their UNIX skills on a PC at home. We've had students who have gone to a store in the evening, bought a copy of Linux, and installed it on a spare machine in class the next day.

Marjorie: What do you find most attractive about Linux?

Chris: Learning Tree's main reason for using Linux is not, for the most part, because it is free. More important is that it is easy to install. The company has over 2,500 PCs in classrooms around the world and at the start of each week, virtually every single one gets reloaded from scratch with the operating system, applications and software needed for that week's course.

Dave O'Neal, who runs one of the company's three development labs, comments:

> We've put huge amounts of effort into finding ways to load software onto machines quickly and reliably. It wouldn't be so bad if all our PCs were the same, but they're not. The bottom line is that once the software is loaded and the machine rebooted, it *has* to work. There isn't time for manual reconfiguration or firefighting. We find that Linux is extremely good at figuring out what hardware is out there, loading the right drivers and booting successfully.

Marjorie: Tell us more about what the class offers.

Chris: The course with Linux as its central focus has an explicit mission of promoting the use of Linux within a commercial environment, and we target it specifically for people who need to support Linux in business. In the course, students learn how to install and configure Linux, download and install software from the Internet, build kernels, provide FTP, web and mail services and even how to interoperate with Microsoft.

Learning Tree is a totally independent supplier of training, so we retain the freedom to mention the bad as well as the good things about the products and

technologies we teach. Linux consultant Phill Edwards, who authored the Linux course, recalls an incident at the beta test stage:

> Some of the other members of the development team were playing around at the back of the room and discovered that it was rather easy to start from the LILO prompt and obtain a root shell without giving the root password. It was rather gratifying to be able to not only point out the loophole, but show students how to fix it.

Marjorie: Any parting words?

Chris: Of course there will always be those who are happy to learn Linux by sitting down with the product and a book and playing with it. It is a great way to learn; I originally learned UNIX the same way—except in those days, UNIX wasn't really a product. Now that I think of it, books weren't available either—that works fine if you have the time. However, if you need to get up to speed fast, intensive training is a very effective way to do it.

Learning Tree courses are offered in the USA, Canada, UK and elsewhere. Our web site at http://www.learningtree.com/linux has detailed course outlines, prices and scheduling.

Marjorie: Thanks for your time.

Archive Index Issue Table of Contents

Advanced search

LinuxPOS

Brian Walters

Issue #59, March 1999

One of the strongest features of this program is its client/server nature.

- Manufacturer: Linux Canada, Inc.
- E-mail: sales@linuxcanada.com
- URL: http://www.linuxcanada.com/
- Price: $199 US per lane
- Reviewer: Brian Walters

Being a Linux consultant by night and a Windows programmer by day, I am always looking for the best way to solve different problems. Recently, I was given the chance to apply a Linux solution in a new market for us. A longtime Linux client asked me what I'd recommend for a point-of-sale system. Of course, I knew if he was asking me, he probably wanted a reliable solution based on Linux. After a little searching on the Web, I found LinuxPOS from Linux Canada, Inc. Their web site said LinuxPOS was still in beta but a demo copy was available, so I downloaded it. The beta installed easily and in a few minutes I saw the POS login screen. I was impressed immediately.

One of the strongest features of this program is its client/server nature. Once the registers are up and running properly, the server can actually be taken off-line for backups, maintenance, hardware upgrades, etc. When the server is back on-line, the registers synchronize with the server and the cashiers never notice a thing. This type of configuration makes this application great for remote stores where phone lines may not be 100% reliable.

LinuxPOS is the front end to a powerful Unidata-based program called Retailer's Choice ($599 US), also from Linux Canada. (UniData Server, $885 for a 3-user license, can handle up to six lanes of POS.) RC can handle all the accounting and inventory needs for a single store as well as multiple locations. Written as a text-based application, RC is easy to navigate and simple to understand. You can access it through an xterm window, the Linux console or a terminal such as a PC with a modem. The report writer is a real plus for managers who want to query the system for specific reports not already available in RC.

Screen Shot

The server I set up to run the Unidata database is a Pentium II 400 with 128MB of RAM and two 6.4GB Fujitsu hard drives, housed in a RAID1 full bay enclosure from MicroPal Corporation (http://www.MicropalCorp.com/). Backups are handled by a Seagate DAT drive connected to an Advansys SCSI card. Red Hat 5.2 with iBCS re-compiled handles the OS needs, since the database is not yet native to Linux. Network connectivity is handled through a 10/100 Intel Pro NIC.

The registers also run Linux, of course. AMD K6 266 processors with 32MB RAM and a 2GB hard drive make up the registers. Symbol scanners and Star printers were added to handle the specific POS needs. I believe it is very important that any Linux system sold includes some sort of reliable, fast backup and restore process. These registers are no exception. With a floppy disk in hand, a store manager can walk up to a crashed register and restore it in five minutes while typing only one command. This type of restore is possible due to the 100T network and etherboot.

Another great feature of this system is its technical support. Beside the on-site support provided by myself, the folks at Linux Canada were always there, ready to solve any problems we couldn't quickly fix on our own with the help of the manuals. Support can be a tough thing for small companies to provide, since so much effort is consumed in actually writing and enhancing a great program—these folks have found the perfect balance.

While it appears that LinuxPOS can do everything a storekeeper would want, there was one place where I had to find another solution—employee time cards. Sure, you can have them punch a card in and out every day, but with all

this processing power, there is a better way. At first I thought I had found it through a great little application called TimeClock, but after contacting them, I discovered they are having internal difficulties and could not provide us with what we needed. So what's a good Linux consultant to do? Simple; I wrote a series of scripts called lxTimeClock (http://www.TexasComputers.com/) and released them under the GPL, free to all. Support is very limited, but once you get things installed, it really doesn't need much attention at all.

With a little ingenuity and a few years of Linux experience, I was able to deliver a strong, reliable solution without much custom programming. Many people get stuck on how free Linux is and toss a computer together from old or cheap parts. This is fine for testing it out at home or in your office, but when it comes to delivering solutions, quality parts make all the difference in the world. So the next time someone asks you “Can Linux do...?”, search the Web—the answer is most likely there. LinuxPOS was just sitting out there waiting for a great opportunity like this one.

Brian Walters (Brian@TexasComputers.com) has worked with computers since the early 80s. He fell in love with the Macintosh in 1990 and became a UNIX enthusiast in 1996 when introduced to SCO and Linux. At that time, he formed R & B consulting to specialize in providing unique solutions for small and medium businesses using both Linux and Windows. He enjoys hunting in his Jeep, but doesn't like to get too far from society, as he cannot live without the Internet.

Archive Index Issue Table of Contents

Advanced search

Red Hat LINUX Secrets, Second Edition

Duane Hellums

Issue #59, March 1999

I highly recommend this book as a powerful addition to any Linux system administrator's technical library.

- Author: Naba Barkakati
- Publisher: IDG Books Worldwide, Inc.
- Price: $49.99 US
- ISBN: 0-7645-3175-1
- Reviewer: Duane Hellums

Red Hat LINUX Secrets, Second Edition is well-written and an improvement over the first edition in many ways. The bundled CD, Red Hat Linux version 5.1 with Linux kernel 2.2.35, offers several enhancements over the Slackware distribution provided with the first edition. I highly recommend this book as a powerful addition to any Linux system administrator's technical library. *LINUX Secrets* is geared toward the intermediate to advanced reader. Novices should consider a more straightforward and less complex manual, such as *Teach Yourself Linux in 24 Hours* by Billy Ball from Sams Press, which comes with version 5.0 of Red Hat Linux.

LINUX Secrets is organized quite nicely into four sections and two appendices. Chapters 1 and 2 explain installation and configuration. Chapters 3 through 8 cover basic software applications and operations. Chapters 9 through 17 give detailed information on hardware. Chapters 18 through 23 contain advanced topics related to using Linux in business. Appendix A is a rundown of graphical X programs included with the distribution. Appendix B is a very handy list of frequently used Linux commands, with appropriate command line parameters and options. Overall, this organization works fine, although the reader may want to take a look at Appendix A after reading chapter 2 and before proceeding to chapter 3. Also, the reader may find it more logical to transpose Dial-Up Networking (chapter 18) and Tcl/Tk Scripting (chapter 8).

I like Naba Barkakati's style of writing. It is highly technical without sounding that way and is very easy and interesting to read, even for the newcomer. He makes good use of margin icons and cross references to related concepts and tasks. The index is thorough, precise and highly efficient.

LINUX Secrets contains pertinent, comprehensive, timely and accurate information. Naba Barkakati covers all the main topics and misses little. I did notice some mismatches; however, they were minor and appeared to be primarily by-products of switching from Slackware to Red Hat, and of the decision to include a newer version of the operating system kernel (2.2.35) than described throughout the book (2.2.32-34).

The CD comes with all the necessary tools to build a basic Linux workstation or an industrial-strength work group, Intranet or Internet server. This includes a plethora of useful software applications such as the X Window System, window managers styled after Motif and NextStep, games (Chess, Galaga, Backgammon), Midnight Commander file manager, text editors, graphics and paint programs, spreadsheet, music CD player, Apache web server, Netscape Navigator, DOS/Windows connectivity tools (Samba LAN Manager, VFAT support), communications programs, C/C++ compilers, TCP/IP networking, a suite of applications (PPP, POP3 mail clients, TELNET, FTP, DNS, NFS, Sendmail, news servers and readers) and power tools such as GNU Emacs, Perl, Tcl/Tk, X programming and RCS. IP masquerading in an office or home LAN configuration even enables a Linux box to function as an effective Internet router at a fraction of the cost of competing technological solutions.

Red Hat has done a great job of streamlining, automating and “dummy-proofing” the installation process. They created a graphical installation program that runs by default on booting from the created boot disk. They also improved maintenance prospects by including the Red Hat Package Manager (**rpm**) to facilitate installing, uninstalling and verifying installation of add-on software applications.

Red Hat Linux used “autoprobing” to automatically configure the majority of my hardware devices. I was up and running in less than an hour, playing the Galaga video game in X and accessing resources on my local area network (I rebooted only once). The X server is much improved over the GUI distributed with Slackware and comes with a nice assortment of flexible window managers. Mr. Barkakati efficiently walks the user through a typical installation. His instructions are comprehensive, detailed where necessary and well-written and organized. I found this version of Linux to be easier, cleaner and more efficient than Slackware. I get none of the error messages or slow bootup processes common to some earlier versions. Countless modifications and tweaks have been included by Red Hat with this distribution, making it a real joy to use.

However, this doesn't mean Linux installations are finally ready for prime time. I think a more flexible and robust installation program would serve users better in the next iteration of Red Hat Linux. Specifically, the Druid **fdisk** partition tool “alternative” doesn't function as well as it should with second hard disks. Also, the LILO boot manager doesn't provide the option to initially install on a floppy disk to prevent writing over NT, OS/2 and Windows 95 Master Boot Record information.

It would be nice to see some extra CD goodies included, such as Doom and Quake which are freely available elsewhere. Also helpful would be a loadable module for sound card support to avoid having to manually configure and rebuild the kernel (and install the kernel source files, which aren't loaded by default). The assortment of arcane and unfriendly text editors leaves a lot to be desired—Linux is in dire need of an intuitive, commercial-quality, freeware, GUI-based word processor (I typed this using vi and the Pine mail reader—Wordpad for Linux, anyone?).

The time and effort required to get a Linux system up and running, connected to the Internet, printing, sending e-mail and accessing multimedia content is still way behind the power curve compared to other web-optimized, user-friendly operating systems. These things must become ubiquitous before Linux can compete as a viable desktop operating system for the masses (complain all you want about Bill's market influence and control, but he gave us exactly what we want and need: an easy-to-use GUI that is multimedia-capable and Internet-aware, with an operating system and application installation paradigm we can all live with).

LINUX Secrets would benefit from more detail on using the X GUI-based Control Panel configuration tools (kernel, LILO, modem, networking, users/groups, etc.). Also, some of the old Slackware-documented methods should be standardized and made consistent with Red Hat Linux.

All in all, this is a great book and well worth the price.

Duane Hellums is a program manager, software engineer and IT consultant, with a Master of Science in Information Systems degree and eight years of network and system administrator experience. He can be reached at hellums@zebra.net.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #59, March 1999

TkZip, tkWorld, tknotepad and more.

Since the debut of this column, I have received several e-mails with suggestions for software packages to review. Last month, I looked at a few packages built with GTK+. While these still seem to be the large majority of start-up packages, ones written in other languages are also available. I will highlight one or two that use Tcl/Tk. It is also true that not all the good programs are graphical.

TkZip:http://www.pcnet.com/~proteus/TkZip/TkZip.html

TkZip is a very well done program that can handle just about any compression utility available on Linux, including tar, bzip, bzip2, gzip, compress, cpio, zip and more. It will search your system for the utilities it needs. The interface is intuitive and aesthetically pleasing. If you are uncomfortable with archiving or compression utilities, either creating or extracting them, this program should be high on your list of “must haves”. It requires Tcl/Tk 4.2 or higher.

tkWorld:http://www.tkworld.org/

I have looked at several very nice GTK utilities, such as **gtkfind**, that did an excellent job and were intuitive to use. The one shortcoming I noted was that the command which actually does the work could not be captured. tkWorld allows you to see the command line and capture it for use in a script, as well as see the results in a window. tkWorld also allows you to create complex “scripts” by selecting a pipe and continuing with other commands. Currently, you can pipe **find** through **grep**. While I don't consider the interface quite as appealing as the interface in gtkfind, this program shows a good deal of promise, particularly when the “Registry”, the list of utilities tkWorld recognizes, is better filled out. It requires Tcl/Tk 8.0.

tknotepad:ftp://ftp.mindspring.com/users/joeja/

Just migrated from Windows to Linux? Still use Windows and its note pad? Well, **tknotepad** will have you feeling at home. Apart from the grey background, it looks and feels exactly like the note pad from Windows, complete with a selection for turning "Word Wrap" on or off. This gives all those who fear vi, but don't want the complexity of Emacs, reason to rejoice. It requires Tcl/Tk 8.0.

dut:dkampman@xs4all.nl

The premise for **dut** is that novice users should not be confronted with a command line while in a GUI. Buttons present the user with choices for selecting the file to work with, highlighting the file and editing it, removing, executing, etc. Not all buttons are functional at this time. According to the author, the goal is eventually to be able to replace an xterm for those doing rather vanilla things like creating, editing and executing files. The only problem I encountered was the same one most new users who don't read English encounter: I was not always sure what the button was supposed to do. A good initiative to keep the Linux momentum alive would be to allow users to set a language preference so that buttons, boxes and dialogs would be understandable to them. It requires libc5, xforms-0.88, X11 and libm.

compjuga:http://csg.uwaterloo.ca/~dmg/compjuga/

compjuga provides a complete conjugation of whatever Spanish verb you specify. Currently, it is a command-line-only program. Writing a GUI wrapper should be trivial, although I don't know if the author is considering that as an option. A good compromise would be a GUI interface if the DISPLAY environment variable is set, otherwise output as it does now. The author does promise a move away from gdbm. It requires glibc and gdbm.

GNU Pilot LogBook Pro:ftp://ftp.stampede.org/skibum/

Pilot LogBook is a full-feature, functional log book for aircraft pilots. Just about any field you might need is here. You can view totals and make notes. One area that has not yet been implemented, but is planned (if you believe in labeled buttons), is Medical Info. For me, not much is missing. I would prefer the totals included categories for 90 days and six months back, common FAR prerequisites for certain flights. The database is a flat-file database; I would have preferred MySQL. It requires GTK-1.0.6, Xext, X11, libm and glibc.

EReminders:ftp://allie.alliedtours.com/pub/EReminder/

I tend to stay quite busy, but even when on the road I check my e-mail. EReminder has a lot of promise for those who need polite reminders by e-mail. Now, instead of remembering to e-mail myself a note, especially if it is a

recurring requirement, I can let my system do it for me. This is a first release and a little rough, at least in the security area. The author has not yet taken advantage of MySQL's ability to encrypt passwords, so they are stored as plaintext. It requires MySQL, Apache with the php3 module compiled with MySQL, cron and a mail transport agent.

sxid:ftp://marcus.seva.net/pub/sxid/

sxid is great for system administrators who want to monitor the status of **suid**/**sgid** programs. It can be run from cron and the results mailed to anyone who needs the information. Another great security tool to compliment **tripwire** and others. It requires glibc; cron and a mail transport agent are optional.

That's it for now, even though many more good packages are available on the Web.

David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When he's not working, he can be found hacking his own system or enjoying the view of Seattle from an airplane.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux in a Public High School

Andrew Feinberg

Issue #59, March 1999

High school student brings Linux and the Internet to his fellow students.

Walt Whitman High School is located in Bethesda, Maryland, a cozy suburb of Washington, D.C. On the surface, Whitman looks like any other school. However, if you look to your right upon entering the building, you will see something rarely seen in our schools: a neat row of terminals, each displaying the message shown in Listing 1. What is unique about this login screen is that the server it comes from is entirely student operated and maintained. The story behind it is as unique as the situation.

In 1994, the students at Walt Whitman High School were told they were on the very bottom of the list to become part of Montgomery County Public Schools' *Global Access* program. Global Access is an initiative to give all of MCPS state-of-the-art computers and high-speed Internet connections. Having moved into a brand-new building just two years before, Whitman was given low priority for the technology upgrades. However, the community did not stand idly by and allow their school to become a technological backwater. With community support, the school purchased three Power Macintosh computers which are now used for multimedia presentations and a few Windows 95 machines for the WWHS Media Center. These computers were put in a storage room off the

Media Center, creating the “Tech Team” where students could use the few available advanced computers for projects and research. However, some students had an even more revolutionary idea: use Linux to provide e-mail accounts and text-based Internet access to everyone in the school.

George Mason University's Center for the New Engineer had a grant to provide several schools in the Washington, D.C. metro area with ISDN links to their four T1 lines. Walt Whitman was chosen as one of the schools. The school already had a donated Compaq Proliant server with the following specifications:

- Intel Pentium 100 processor
- 48MB of RAM
- 4.2GB of disk space with Compaq's HotPlug system
- DigiBoard IMAC ISDN router
- Lucent/Livingston PortMaster 2e

Although the machine originally had Novell Netware 4.1 installed, the Tech Team decided to use Linux as the server's operating system. The original system administrator, Gerald Britton, spent hours setting up the server and adding accounts for 1500 students and staff. The server now has over 2000 users and handles a large volume of mail. It is maintained by a team of five students and hosts the school's web site (http://whitman.gmu.edu/) which receives several thousand hits a week. The web site is used for such things as school announcements, contests and even homework postings by teachers.

Five years later, Whitman students still use the server for research projects, to keep in touch with mentors, friends and family, and to communicate with people all over the world. To facilitate easy use of the server, the Tech Team installed a menu system for students and staff to use so they don't have to explore the innards of Linux unless they truly want to. However, many students elect to use a conventional shell and have learned real-world UNIX skills on the server.

Whitman has a one-channel ISDN line to George Mason University which connects their entire school network to the Internet pending the arrival of Global Access. However, the arrival of county support may not be good news for the Tech Team. The county has not given the Tech Team permission to run their server on the Global Access network. Therefore, the Tech Team is currently searching for an Internet service provider to donate bandwidth so they can continue their student-run Internet services. Their system now boasts a Livingston PortMaster 2e and 25 USR modems attached to phone lines throughout the school, with which the Tech Team is experimenting with plans to allow students to dial in and check their e-mail from home at night. Walt Whitman is one of a few public schools using Linux and currently one of two

schools in Montgomery County which provides e-mail to all its students. The other school plays host to a Computer Science magnet program that only recently has opened its system to the whole school. MCPS has an e-mail system called FirstClass that students can use. However, students applying for an account must first find a staff sponsor before they can submit their application. Whitman requires only that the student and the student's parents sign a contract promising that the student will not misuse the system. Despite fears that students would use their accounts for non-educational purposes, they understand their access is a privilege and each one of them has a responsibility to safeguard it for the school.

With the Internet becoming more of a research tool than a toy in schools today, it is important that all students have equal access to services like e-mail. Whitman has tried to level the playing field a bit and allow all of its students to have the same opportunity to communicate and share ideas. Whitman's system administrators hope they can find enough resources to continue the service. Whitman's use of Linux is only a small part of the picture. The focus of their efforts isn't just to promote Linux in schools, but to provide the rest of the school with a valuable learning experience that shows the Internet to be a useful tool in the real world.

At the time I originally wrote this article, Whitman's connection was in the process of being severed. The Tech Team submitted proposals to local corporations in the hope that they could receive enough funding to continue the service. Now, Whitman has received a grant from Science Applications International Corporation (http://www.saic.com/) and Bethesda Online Organization (http://www.boo.net/) that will allow it to continue its Internet access for another year.

Andrew Feinberg is a student at Walt Whitman High School. He is a part-time developer for Debian GNU/Linux. He can be reached at andrew@ultraviolet.org.

Archive Index Issue Table of Contents

Advanced search

The Linux Router Project

David Cinege

Issue #59, March 1999

A look at one of the fastest growing Linux distributions, that you may never actually see.

If you build it, they will come... The following definition can be found at http://www.linuxrouter.org/; however, LRP needs a little more thorough explanation than this:

> Linux Router Project (LRP) n. A networking-centric mini-distribution of Linux. LRP is small enough to fit on a single 1.44MB floppy disk, and makes building and maintaining routers, terminal servers, and typically embedded networking systems next to trivial.

LRP began because of my intense desire to learn Linux. After mutilating an installation of Slackware and actually installing Debian 1.0, I finally had a clue as to what was going on. I planned to dive deep into TCP/IP networking and decided I needed a dedicated Internet connection for which I would need a router. Of course, I could have *bought* a router, but that would have been too easy (beside, I am too cheap). So I said, “Hey, let's build one. That will be a great way to waste countless hours—oops, I mean learn Linux.”

In my travels for information on building a router with Linux, I could not actually find anyone who made what I had in mind. Either they were using a full distribution on a hard drive, or something on a floppy that was so minimal a second complete machine was needed to configure the disk. I decided to try to make this self-contained “mini-system” myself. I started with a minimal base install of Debian 1.1 which totaled around 35MB and began slashing (and fixing what I had broken) until it was down to around 3.2MB. This was small enough to fit on a 1.68MB floppy as a gzipped initrd image. It was not pretty, but it did the job as a router for my 33.6K line, which eventually progressed to ISDN.

After several months, I wanted to take what I had made and create a more generic system for other people to use. I put out the call on Usenet and a few private lists, then began a mailing list with a handful of interested people. The rest, as they say, is history. That mailing list now exceeds 600 names, and at least 25 new downloads of LRP are done each day. This is all with LRP still remaining mostly unadvertised.

My single-disk hack job has evolved into a modular micro-sized operating system. It makes a capable base platform for just about anything you could want to build for “light”, secure, efficient use: routers to terminal servers, mini web servers to DNS cache and even doorbells. (No, I am not kidding.)

LRP Defined

First, Linux Router is quite small. The base root file system is only 2.5MB and compresses down to 830KB. With the addition of a kernel, it easily fits on a 1.44MB floppy with several hundred kilobytes to spare for kernel modules and additional packages. A Zip or LS-120 will provide seemingly limitless space. Flash RAM can also be used and companies like Sandisk and M-Systems make Linux-capable products.

LRP's root file system is contained in a gzipped tar archive. The directories /etc (etc.lrp), /var/log (log.lrp) and /lib/modules (modules.lrp) are broken out into their own archives for easier backup. At boot, the Linux kernel dynamically creates a MINIX file system on /dev/ram0 and mounts it as / (root). It then extracts the root archive (root.lrp), storing it in the / directory, then proceeds to execute **linuxrc**. linuxrc sets up the base root file system (like creating /dev) and then attempts to mount the boot device and install any remaining packages (such as etc.lrp, log.lrp, and modules.lrp). The remaining bootup is noticeably similar to a typical full-sized Debian Linux install.

The system will be running solid state from the RAM disk. This means all changes you make will have to be backed up to the boot medium. This is not a difficult task, as it is all automated by scripts that can recreate the root and add-on packages from package listing files found in the /var/lib/lrpkg directory. **lrcfg** is a menuing system that gives you a nice interface to the package backup system as well as configuration files and (minimal) on-line help.

Note that LRP does *not* have to run from RAM. It is perfectly fine just to dump the root and package archives to a hard drive partition and bypass the “embedded” RAM disk features.

Included in LRP

If you should drop to the command line, you will find ash as your default shell. ash supports full Bourne shell scripting, and nearly the entire LRP base is fully customizable shell scripts. Moving around at shell level, it will be hard to tell you are not using a full distribution.

Some of the finer details include complete SysVInit, cron, libc6 (glibc) and sed. The system and binaries are generally kept up to date with the current Debian release. Over 60 commands are available in minimal or emulated form (using ash and sed): cat, cp, dmesg, dd, grep, cut, passwd, gzip, tar, tail, mnc (mini-netcat), et al. Of course, all standard and many extended networking commands and daemons are also available: route, netstat, ipfwadm, ipautofw, ipportfw, inetd, telnetd, tftpd, routed, et al. Anthony's Editor (ae) serves as your editor.

If this is not enough, you can also load Secure Shell, pppd 2.3.5, Portslave RADIUS client, gated, snmpd and others. Add-on packages such as minicom and bash 2.0 are also available. Add-on packages are installed once and merged with the root archive at back-up time, unlike full packages which are kept autonomous.

Slackware users can stuff in almost anything else by hand. Remember that LRP is *real* Linux—just smaller. It is capable of running anything a full-size distribution can run, given the libraries and space.

Uses of LRP

By design, LRP is meant for low-level networking applications. To this extent, the stock Linux 2.0 LRP kernel is compiled with almost all networking options and several enhancement patches (for example, ipportfw). Next is a short list of default functionality available in the base. Understand that these features apply to any interface type Linux can use, from modem to wireless T1.

- Full IP routing with numberless IP and multiple IP interfacing
- Complete firewalling and IP accounting
- Traffic shaping (low-speed links)
- IP and Port (TCP/UDP) redirection with transparent proxy and IP load balancing
- Extended Common NAT (IP Masquerade) as firewall rules
- Media insensitive interface load balancing (EQL)
- RIP, OSPF, BGP and other routing protocols (via gated and other packages)
- IPX and AppleTalk support

Past these core functions, a Linux Router unit is quite viable for many server applications. One of the most popular is using the Portslave RADIUS client software (pslave.lrp) to host inbound Internet access modem pools, which give you functionality like the popular Livingston Portmaster series. Bind is packaged (bind.lrp) and does a fine job as a secondary or even primary DNS. Boa (boa.lrp) and thttpd (thttpd.lrp) are both small web servers that will fit on a 1.44MB LRP disk. Apache, though not packaged right now, could be hand fit.

On the client side of things, the VNC package will give you a single-disk Windows terminal. XFree86 is not yet packaged, but it certainly will be, allowing easy creation of X terminals. Linux Router also makes a good base for workstations that mount their file system from remote NFS or SMB (Samba) servers.

Understand that Linux Router is not the solution to all your computing needs. You wouldn't want to use it as your primary web server if you are an Internet Service Provider, or as a kernel hacker's development machine. But for backup services or where feature requirements are very specific (and administration skills scarce), it makes an efficient and sound choice.

The Advantages of LRP

While a “minimal” install of Debian Linux may have well over 5000 files, you would be lucky to break 500 with a very feature-rich LRP. The impact of this on administration and security is obvious; the entire system can be backed up and restored in less than a minute.

This minimal footprint coupled with RAM disk operation adds up to a large performance increase in operation and decrease in required hardware. The most mediocre 16MHz 386SX with 8MB of RAM, a 1.44MB floppy and no video card can handle most people's needs for light routing jobs. Few will have a need to climb past a 486 class CPU; however, with low-end Pentium class machines being so inexpensive, it certainly makes sense to do so.

Furthermore, with this default RAM disk approach, it is very difficult to be left with an unusable system. File system get corrupted? Flash the power. Everything just crashed? Flash the power. Cracker break in? Install your backup (in 30 seconds) and flash the power.

The other advantage of running from a RAM disk that people seem to overlook is, generally, it *cannot break*. Let's say you have 250 workstations with one path to the outside world. Do you want that path to depend on a mechanical device like a hard drive? Probably not. How about if that router sits between you and your game of Quake? Definitely not.

Some of the many reasons to use LRP over other Linux systems: fully administrable via high-strength encryption using secure shell and secure copy (ssh, scp), a huge routing feature super set and open source. These are things most commercial solutions cannot touch.

Locating LRP

By now, I am sure you are probably excited to give Linux Router a try. Since it is now easier than ever to make a disk, you should do it.

All that needs to be done to get started is creating a file system and installing a boot loader on your boot medium. The most common choice for this is MS-DOS (FAT) and Peter Anvin's SysLinux. GRUB also makes a good boot loader. LILO is generally a poor choice, because unlike the former two, it cannot dynamically search for the kernel by name, and LRP does not have LILO available if you change kernels. Default syslinux.cfg and syslinux.dpy files are available at the LRP ftp site.

An LRP-capable kernel can be found in the kernel*.tar.gz archive. Choose an FPU (floating-point unit) or non-FPU kernel as needed and copy it to the disk as “LINUX”.

Copy the base and any additional packages; root.lrp, etc.lrp and log.lrp are required. modules.lrp is also needed if you are using a kernel with modules, as the provided LRP kernels do.

Finally, you must update the options passed to the kernel at boot time, commonly known as the boot loader's “append” line. A sample syslinux.cfg file looks like this:

```
display syslinux.dpy
append=load_ramdisk=1 initrd=root.lrp \
initrd_archive=minix ramdisk_size=4096 \ root=/dev/ram0
boot=/dev/fd0,msdos \
LRP=etc,log,modules
```

The options after **append=** will be common to any boot loader you use. The LRP-specific options are:

- **initrd_archive=minix**: initrd_archive is a kernel feature that lets you use tar.gz archives instead of raw images. This kernel patch is required in order to use LRP.
- **boot=/dev/*bootdev*[,*fs*]**: this is the device name you are booting from. It is the device which linuxrc will try to mount to install any remaining packages. Optionally, you can specify the file system for the mount attempt. The boot line is semi-optional. linuxrc has a back-up list of devices to try to mount. It looks at /proc/filesystems for available file

systems. Using **boot=** speeds up the boot process and ensures a mount if you are using an odd boot device.

- **LRP=etc.lrp,log.lrp[,modules.lrp][,package1] [,package2]...**: these are the packages linuxrc should try to load at boot time. Remember that root.lrp is loaded by the initrd facility of the boot loader, so **etc** and **log** are the first packages listed here.

If your goal is simply a 1.44MB LRP floppy, the FTP site contains a raw "idiot-image" of such a disk. It is fully prepared with syslinux and a non-FPU kernel; just use dd, cat or a similar utility to write the image to a floppy.

With your base prepared, you must add the needed kernel modules. Normally, one does this by booting the LRP disk, mounting a second floppy, copying the modules from that floppy to /lib/modules, editing /etc/modules and then using lrcfg to back up the modules package. Sound too hard? See http://www.linuxrouter.org/modmaker/ for a modules generation system. Just select the modules for the features and hardware you want; it will create a modules.lrp with the modules, their dependencies and an /etc/modules file configured to load all of them.

After boot and login, lrcfg will be started to help you get around to the files you need to edit. The configuration files for the base and packages are self-documented. You should be only a few minutes away from a working system.

Difficulty and Support

For a person with some UNIX networking experience, LRP is truly as easy as it sounds. However, it may be difficult for people who lack these basic skills. Still, it is not too intimidating and the average Linux, MS-DOS or MS Windows user has been known to tackle the entire task with no help.

The mailing list is capable of getting most people over any walls they encounter. If all else fails, both Paul Wouters (paul@xtdnet.nl) and I currently provide commercial support for those requiring extended help with their setup.

Commercial Products

Linux Router is known to be in use around the world. I know of several consultants who use it exclusively for networking their customers. It is also becoming popular with vertical applications and has been spotted in control systems and power switching stations.

Onyx Systems (http://www.onyxsys.com/) is developing a mid-range modular router and terminal server product based entirely on LRP. Look for it to appear about the time this article is published. This is the most adventurous

application using Linux Router I have found (sort of a cross between a Cisco 2524 and Portmaster 3). It is also the first open-source product of its kind—how exciting! Keep it in mind when you need trusted hardware with commercial support.

Rumors are circulating that Corel has been thinking about a port of Linux Router to ARM for a FlashRAM-only NetWinder. This project sounds like it has many possibilities as well.

Development

At the time of this writing, 2.9.4 just went out as an unstable release, getting us a bit closer to a stable 3.0 release. I still handle all of the core LRP development myself and could use more people to help speed development of the base. Making LRP packages is quite easy, and I would like to see more people contributing them. If you have the skills and are interested in helping out, join the mailing list (linux-router@linuxrouter.org).

Resources

Dave Cinege (dcinege@psychosis.com) is an Electronics and Computers Engineer. He lacks anything even remotely resembling a social life. When not hacking (which is rare), he is generally reading technical books, spook lore or arguing the virtues of anarcho-capitalism. Aside from qualifying as a truly pathetic individual to the uninitiated, he is one of the most rounded jack of all trades you may ever find.

Archive Index Issue Table of Contents

Advanced search

Cost-Effective Services for the Office

Kim Henderson

Issue #59, March 1999

How the Linux operating system made possible cost-effective company e-mail and created opportunities for adding useful services.

Aerofil Technology had been using cc:Mail to support internal e-mail needs for about a year and a half when I joined the organization in 1995 as an Information Services Technician. The IS manager at that time had implemented cc:Mail along with a major upgrade of computers to Windows 3.11. Networking was installed to allow connections to a Novell server for file and print services as well as access to the corporate accounting and manufacturing software.

Around my first year, the computer inventory, user base and cc:Mail usage grew, requiring the purchase of additional licenses. In 1996, my boss left to do independent consulting and I was given the opportunity to run Information Services. A one-person department has its advantages; in particular, I could make all decisions regarding technology. It also has its disadvantages, such as limited funding for projects and necessities like software licenses.

During this time, our local phone company began providing Internet service and Aerofil signed up for the corporate option, including 20 e-mail accounts which were quickly issued to a privileged few. No grandiose plan was in place with the initial sign-up, but once a few employees had Internet e-mail, everyone wanted it. Also, more and more of our customers wanted to communicate in this way. Having the ISP maintain 70+ e-mail accounts would obviously be costly, and we would not have control over the accounts. The ability to change passwords was important. So, I asked for help from our ISP in obtaining a domain name and in hosting that domain for the company.

Because so many users required access to e-mail, I set up a Linux machine with **diald** and IP masquerading and named it “Gatekeeper”--an Intel P133 with 32MB of RAM. This machine was a vast improvement over the previous configuration in which certain individuals had their own modems, plus we had

a modem server that didn't allow more than one person at a time to access it. With Gatekeeper running, everyone had equal access. Whoever accessed it first would initiate the dial-in, but once connected, everyone had instantaneous access. Typically, once I had my e-mail client running, the connection was up all day. This was not a problem for our ISP, since their heaviest usage occurred between 8 PM and 10 PM. We were on-line only between 7 AM and 5 PM.

With the Linux machine in place, I investigated setting up **fetchmail** so that we could handle all of our own e-mail accounts. At the same time, I also began looking at ways to gain more control of our web site maintenance. Our ISP required that any changes be e-mailed to them for implementation. They eventually set up a configuration to do this using Samba, but it was still troublesome due to name-mangling problems.

Once our ISP began offering DSL (Digital Subscriber Line) service, I decided to wait until DSL could be implemented before pursuing direct e-mail and web site account management. Our IS budget would not allow us to implement DSL as a solution until January 1998.

In the meantime, in order to enable TCP/IP on our network, I set up a second Linux machine, appropriately named IPkeeper, to act as the DHCP (Dynamic Host Configuration Protocol) server—an old Intel 486SX33 with 16MB of RAM. This allowed us to specifically assign IP addresses by hardware address or simply from a set range of addresses. I set this up on a separate Linux machine in order to simplify the Apache configuration on Gatekeeper and to segregate the Intranet from the Internet.

In January of 1998, I placed the order for the DSL connection and added an additional Ethernet card to Gatekeeper for the DSL equipment connection. By February, Aerofil had a 128K connection to the Internet. Testing showed the connection speed to be adequate for hosting our web page. I then notified the ISP to make the necessary changes to DNS to reflect our domain location at Gatekeeper. I moved all of our web pages down and set up all the e-mail accounts that had existed at our ISP, as well as adding additional accounts for the other users who required e-mail. At this time, the company drafted an Internet and E-mail Usage Policy to discourage inappropriate use of the service. In hopes of conserving bandwidth, Internet access was restricted to those users with a justified need. This was accomplished by modifying the IP masquerading rules on Gatekeeper to allow access to specified IP addresses.

The Intranet

With IPkeeper on the network, even more opportunities were available to us. Ongoing discussions had been held to decide how to “computerize” corporate documents and make them accessible to every PC on the network. A variety of

software solutions for Windows NT were looked into and found to be too expensive for our company. Instead, I used existing Internet technology and set up IPkeeper to host our Intranet, upgrading it to an Intel PII-233 with 32MB of RAM.

In June of 1998, the company hired a college student for the summer to assist in getting our existing documents on-line. Some of the documents were in a word-processor format, some were flowcharts done in Visio, and some existed only in paper form. Several methods were used to get this information onto the computer: scanning, converting to PDF format and coding in HTML. Basically, I set up various user areas to segregate the documents, such as Human Resources, Information Services, Aerofil Process Descriptions, Quality Control, Safety Process Descriptions, etc. Each area had security set up to ensure that only designated users could add or modify the information contained in it.

Figure 1

Recent Projects

In order to add more functionality to the Intranet, I implemented the search utility HT://DIG on IPkeeper. It has proven to be one of the most useful features I have added. It allows our users to type in any keyword and instantly see a list of corresponding documents. I am also currently evaluating the Cyberscheduler application for calendar and scheduling services.

An LDAP (Lightweight Distributed Access Protocol) server is being implemented for internal use on Gatekeeper so that users who are using Netscape Communicator can get to a company e-mail directory and utilize company mailing lists more easily.

Samba has also been enabled on both Linux machines, and I am setting up home directories in order to lessen dependency on the aging Novell server.

Future Projects

The company has been investigating computer archive solutions for old report data and converting report files from the business accounting and manufacturing package to HTML so that the files can be indexed and viewed with a browser. The plan is to set up a system which will automatically process the report files, eliminating the need to print monthly or weekly reports as they will be easily accessible through the browser. Seth Golub's (seth@cs.wustl.edu) **txt2html** program is being used to convert the text report to an HTML file. The most likely solution for indexing will be to index only the report headers, which contain the necessary information for locating a particular report.

Final Comments

For a mid-sized business such as ours, Linux was an obvious choice. The company would have been years away from implementing all the services we added if Linux had not been available. No other operating system can allow as much scalability and flexibility for its price.

Kim Henderson has been involved in Information Systems for nine years in both financial and manufacturing businesses. She has been involved with her husband, Darrin, even longer and owes much to him for his love and support. Kim spends her free time renovating their 40-year-old home, coordinating activities for the East Missouri Linux Users Group and maintaining a basement computer network with her husband and their dog, Linus. She can be reached at kim@mo-biz.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Creating a Web-Based BBS, Part 3

Reuven M. Lerner

Issue #59, March 1999

Mr. Lerner shows us how to add a full-text search to our BBS.

Over the last two months, we have looked at a simple bulletin-board system that can be incorporated into a web site. This BBS groups messages by subject or thread, and stores the messages in a relational database.

As we have seen in the last two installments of ATF, using a database for information storage and retrieval significantly reduces the amount of development work necessary to implement such a system. Given the ease with which I was able to implement the BBS, I decided to add several more features which can help users navigate their way through and use the BBS.

This month, we will look at how to implement several of these features. The most important is full-text search, which allows users to find interesting postings based on keywords. This saves them from having to search through a thread, which might not have an appropriate or easy-to-understand title. Then we will look at a tool that allows administrators to remove inappropriate postings without having to go into the guts of the database.

In Case You Just Tuned In

If you did not catch the last two installments of “At the Forge”, let's take a quick look at how the BBS is implemented. I used MySQL (see Resources), a relational database that has gained quite a bit of popularity among web programmers. Information in a relational database is stored in tables, in which the rows represent records and the columns represent fields.

We define the columns with SQL, the Structured Query Language that is a well-known standard for working with relational databases. Our BBS will contain two separate tables, ATFThreads (for keeping track of the individual threads, including the initial posting) and ATFMessages (for keeping track of individual

messages). SQL is sent to a database server from a database client; this can be a programmatic client (e.g., a CGI program) or an interactive client (e.g., the **mysql** program that comes with MySQL). I normally use interactive clients for table creation, maintenance and debugging, but we can create our two tables with the following SQL:

```
CREATE TABLE ATFThreads (
id SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    subject VARCHAR(255) NOT NULL,
    author VARCHAR(60) NOT NULL,
    email VARCHAR(60) NOT NULL,
    text TEXT NOT NULL,
    date DATETIME NOT NULL,
    UNIQUE(subject)
)
CREATE TABLE ATFMessages (
    id MEDIUMINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    thread SMALLINT UNSIGNED NOT NULL,
    subject VARCHAR(60) NOT NULL DEFAULT
    "No subject",
    date DATETIME NOT NULL,
    author VARCHAR(60) NOT NULL DEFAULT
    "Mr. Nobody",
    email VARCHAR(60) NOT NULL DEFAULT
    "atf@lerner.co.il",
    text TEXT NOT NULL
)
```

If you enter the above into the interactive mysql program, you will want to put a semicolon (;) after each query to indicate that you want mysql to execute the query right away, rather than wait for additional input.

Once we have created the tables, we will have to write some programs—in our case, CGI programs—that manipulate the data. We will not write programs with SQL directly, but will create it within our CGI programs. Using Perl for such CGI programs is particularly easy once we use CGI.pm (the standard Perl module for CGI programs) and DBI (Perl's generic database interface).

The entire bulletin board system consists of about seven programs, each of which handles a different aspect of the system. You can download the programs from the FTP sites ftp.linuxjournal.com/pub/lj/listings/issue57/3193.tgz and ftp.linuxjournal.com/pub/lj/listings/issue58/3252.tgz Listings for this article will be in ftp.linuxjournal.com/pub/lj/listings/issue59/3296.tgz

Now that we have given the basic functionality a quick discussion, let's start to add some advanced functionality to our BBS.

Searching

The first new function we will add to the BBS is full-text searching. I have long been a fan of full-text search, both on the Web and elsewhere. According to Jakob Nielsen, probably the best-known web-usability researcher, many users are “search-dominant”. This means they prefer to search through a site for

content, rather than traverse through trees of hyperlinks. (See Resources for more information on Nielsen, including the URL of his article.)

While the threaded structure of our BBS makes it relatively easy to find postings on a particular topic, undoubtedly there will be times when subject lines fail to reflect the actual content or when discussions veer into unexpected or unusual directions. Allowing users to search for words or phrases makes it easier for them to find what they want. Best of all, because the search functionality is in a separate program, it slows down the system only when someone uses it. If no one ever searches through the BBS, the system will not be slowed down.

MySQL allows for two kinds of searches through a table, with either SQL regular expressions or UNIX-style regular expressions. SQL regular expressions might seem silly to someone used to working with UNIX, but they are guaranteed to work on any database system that adheres to SQL standards. SQL regular expressions have two special characters: % (which matches zero or more characters) and _ (which matches exactly one character). To escape these special characters, you insert a leading backslash (\). To get a literal backslash, you insert two backslashes (\\).

To search for a match with SQL regular expressions, you use the **LIKE** operator within a **SELECT** statement. This returns all of the rows for which the regular expression finds a match. For instance:

```
SELECT text FROM ATFMessages WHERE text LIKE "a%"
```

will retrieve all message texts that begin with the letter a, followed by zero or more characters.

If you prefer UNIX-style regular expressions, MySQL allows you to use the **REGEXP** (or **RLIKE**) operator, as in

```
SELECT text FROM ATFMessages WHERE text RLIKE "a.*"
```

This will perform the same function as above.

Rather than force one system on all users, I decided to allow for both literal text and UNIX regular expressions. UNIX regular expressions are hard for most people to learn and understand, so the default is to allow for literal text searches. We perform a literal text search by using the LIKE operator, escaping the two SQL regular expression meta-characters with backslashes.

The search form itself (search-form.shtml, see Listing 1 in the archive file for an expanded version) is thus very short:

```
<P>Search for: <input type="text"
name="term"></P>
<input type="radio" name="regexp"
   checked value="no">Literal search
<input type="radio" name="regexp"
   value="yes">Use regular expressions
<input type="submit" value="Search!">
```

This form is submitted to the CGI program search.pl (Listing 2 in the archive file), which performs the actual searching. Search.pl is also fairly simple, although the SQL query is the most complex we have seen in this project. That's because we have to search through ATFMessages to find matches. We also need the message's thread ID number in order to create a hyperlink to view-thread.pl, which allows users to look at that thread.

Creating the Search Query

We perform what is known as a "join" between the two tables, selecting several columns from ATFMessages and one column from ATFThreads. Joins allow us to take only the most interesting columns from two or more tables, grabbing only those matching our criteria. Always remember to set up a relationship among the tables, otherwise you will get the "Cartesian product" of the results, with each of the rows in table A matched up with each of the rows in table B. We thus avoid selection like this:

```
SELECT M.id, M.thread, M.subject, M.author,
   T.subject
FROM ATFMessages M, ATFThreads T
```

which will produce the Cartesian product. Instead, we use

```
SELECT M.id, M.thread, M.subject, M.author,
   T.subject
FROM ATFMessages M, ATFThreads T
AND M.thread = T.id
ORDER BY M.date desc
```

which qualifies the relationship between ATFMessages (given the nickname "M" in this query) and ATFThreads (with the nickname T), then lists the resulting rows in descending date order.

We also test the value of **$regexp**, which is set to the value of the "regexp" radio button. If **$regexp** is "yes", we use the REGEXP operator in our SQL query and perform a regular expression search. Otherwise, we escape the SQL characters % and _ and use the LIKE operator. The Perl code to make this query possible looks like this:

```
my $sql = "SELECT M.id, M.thread, M.subject, M.author, T.subject ";
$sql .= "FROM ATFMessages M, ATFThreads T ";
if ($regexp eq "yes")
{
    $sql .= "WHERE M.text REGEXP \"$term\" ";
}
else
```

```
{
    $term =~ s|%|\\\%|g;
    $term =~ s|_|\\\_|g;
    $sql .= "WHERE M.text LIKE \"%$term%\" ";
}
$sql .= "AND M.thread = T.id ";
$sql .= "ORDER BY M.date desc";
```

Because we build the SQL query by combining text strings, we can conditionally modify parts of the query, as we saw above.

Parsing the Search Results

Results from a SQL **SELECT** query are always returned in a table, in which the columns are the rows requested in the query and the rows are those matching the criteria from the query. With DBI, reading the results from a query usually means iterating through the rows from within a Perl **while** loop.

DBI provides a number of methods for retrieving the values returned by a SELECT, but perhaps the easiest one to understand is the simple **fetchrow_array** method. This method is defined for **$sth**, the "statement handle" through which we submit our query and retrieve its results.

Several methods can be used to retrieve the results from our SELECT, but the easiest one to understand is **$sth->fetchrow_array**, which returns one row from the response. Each time we invoke **$sth->fetchrow_array**, the next row from the response table is returned. After **$sth->fetchrow_array** returns the last row of the response table, it returns "false". By putting **$sth->fetchrow_array** inside of a **while** loop, we can iterate through each of the rows in the response table.

Here, then, is the code from search.pl that iterates through the results table:

```
while (my @row = $sth
{
  ($message_id, $thread_id, $subject, $author,
    $thread_name) = @row;
  print "<li><a href=\"/cgi-bin/view-thread.pl?";
  print "$thread_id#$message_id\">$subject</a>, ";
  print "by $author in ";
  print
"<a href=\"/cgi-bin/view-thread.pl?$thread_id\">";
  print "$thread_name</a>\n";
}
```

As you can see, we assign a number of scalars to the individual elements in **@row**. DBI returns NULL elements (that is, elements that lack a value, rather than the C/Perl notion of "true" being non-zero) as undefined, so you can test for a value with Perl's built-in defined function. Once we have extracted the elements of **@row** into a number of easy-to-identify scalars, we can then use them to print results to the user's browser.

Notice how each of the hyperlinks we create does not simply point to a thread, but also to a message. We can do this by taking advantage of named anchors within a link, which allow us to force the user's browser to scroll to a particular point. If you are unfamiliar with named anchors, here is a quick lesson: in the link http://www.ssc.com/test.html#testing, “testing” is the named anchor and points to a location in test.html marked with the tag **<a name="testing">**. If no such tag exists, adding the named anchor to the URL has no effect.

Because our program view-thread.pl (discussed last month) places such a named anchor at the beginning of each message header within a thread, we can thus point users directly to the message that matched their search string, rather than to the thread.

By the way, if you are interested in getting the greatest possible speed out of your application, you might want to consider using **$sth->fetchrow_arrayref** rather than **$sth->fetchrow_array**. The difference, as you might guess from their names, is that the former method returns an array reference, while the latter returns an array.

Passing a reference will always be faster than passing an array, since it involves manipulating fewer bytes. I chose to work with **$sth->fetchrow_array** partly because it simplified the rest of the code, and partly because I felt that we would be handling small amounts of data anyway and that the speed difference would not be too significant.

With these two files—search-form.html and search.pl—installed on our server, we now have the ability to search through the text of any message. With a few new links to the search form from our main page, this functionality is integrated into our system.

Administration Tools

Whenever I work on a web site that involves databases, I almost always include one or two web-based administrative tools. Such tools are useful for a number of reasons, the most obvious of which is their utility for people who want to manipulate the database without learning SQL. (They also allow you to shield the database from people who think they understand SQL, only to discover there is not any way to undo a DROP TABLE command.)

Which tools are necessary and which functions they must perform will depend on the web application you have written, as well as the needs of your individual users. We will look at a simple application called zap-thread.pl, which allows administrators to delete one or more discussion threads.

In order to ensure that only authorized users can zap threads, we will include a password field, adding the variable *$zap_password* to ATFConstants.pm, the module containing all of our global variables. (See Listing 4 in the archive file.)

While there are a number of ways in which we could implement zap-thread.pl, I found it easiest to write a single program that has two different personalities. When invoked with the GET method, zap-thread.pl produces an HTML form that can be used to delete threads, including a text field into which the user must enter a password.

When invoked with POST, zap-thread.pl assumes it was invoked by the form produced by its GET personality. It expects to receive two different sorts of HTML form elements: a password in the *password* form element, which must be compared with the *$zap_password* variable, and one or more check boxes named "thread-x", where *x* is the ID number of the thread in the database.

Before doing anything else, our program compares the password it received with the *$zap_password* variable. If they match, we continue without comment. If they fail to match, we produce an error message telling the user that the password did not match.

Deleting the Thread

We can iterate through the **thread-x** elements in a number of ways, but I found the easiest way was to iterate through each element, ignoring any elements that failed to match the "thread-x" pattern. By capturing the number within parentheses:

```
next unless ($element =~ m/^thread-(\d+)$/);
```

we can then grab it using the $1 variable, which retrieves whatever was in the first set of parentheses in the last match:

```
my $thread_id = $1;
```

Once we have retrieved the thread ID number, we need to delete matching rows from ATFThreads, which contains the master list of threads, and from ATFMessages, which contains the messages themselves. If we were to delete rows from only ATFThreads, we would run a substantial risk of MySQL reusing the thread ID number with a new thread—which would effectively put all of our supposedly deleted messages in the new thread.

We delete them by sending two separate SQL queries, printing a brief status message to the user's browser:

```
my $sql  = "DELETE FROM ATFThreads WHERE id = $thread_id ";
warn "SQL: \"$sql\"\n";
my $sth = $dbh->prepare ($sql);
my $result = $sth->execute;
die("Error deleting from ATFThreads: " .
    $sth->errstr) unless $result;
print "<P>Deleted the thread.</P>\n";
# Delete messages for this thread from ATFMessages
$sql  = "DELETE FROM ATFMessages WHERE thread = $thread_id ";
warn "SQL: \"$sql\"\n";
$sth = $dbh->prepare ($sql);
$result = $sth->execute;
die("Error deleting from ATFMessages: " .
    $sth->errstr) unless $result;
print "<P>Deleted messages in the thread.</P>\n";
    }
```

By putting the DELETE commands inside of a **foreach** loop, we make it possible to delete more than one thread. If the user indicates that three threads should be deleted, we will enter the loop three times, deleting each thread in sequence.

There is at least one problem with this implementation of zap-thread.pl, and that is the lack of an **undo** function. What happens if you delete the wrong thread? Perhaps the easiest way to implement such an undelete function would be to change the underlying tables, adding a new “active” column to ATFThreads and ATFMessages. This column would contain a single true/false value, perhaps implemented with a TINYINT or an ENUM type.

With such table definitions in place, deleting a thread would involve an UPDATE query (rather than the current DELETE query) to modify the value in that column. Such a change would also require some adjustments to list-threads.pl and view-thread.pl, so that they only SELECT those rows WHERE **active="true"**, or a similar condition.

Another potential issue with zap-thread.pl is that it operates on a per-thread basis. There are undoubtedly times when we will want to delete an individual message, rather than a whole thread. Creating such a program would be a bit more difficult than zap-thread.pl, but the true challenge is in the user interface. Using zap-thread.pl will be difficult enough with 100 threads; trying to find an interface appropriate for zapping individual messages would be even more difficult. The best approach might involve breaking the program into two smaller programs, one for selecting a thread and another for selecting a message.

As with most software projects, this BBS has almost unlimited potential for improvement and expansion. A number of ways remain in which this software could be improved—allowing for sub-threads, providing for moderated discussions, sending e-mail to some or all participants in a thread when a new posting is added, allowing users to edit their own postings and even an integrated spell-checker.

Finally, notice how our web BBS application used separate CGI programs, rather than one single large program. Using a suite of related programs is not the only way to design such applications, but I find it to be the easiest way to create such functionality. Not only does it allow breaking the problem into separate slices, but it also allows for incremental implementation—which comes in handy when a client is breathing down your neck, wanting to see results right away.

Resources

Reuven M. Lerner consultant living in Haifa, Israel, who has been using the Web since early 1993. His book Core Perl will be published by Prentice-Hall in the spring. Reuven can be reached at reuven@lerner.co.il. The ATF home page, including archives and discussion forums, is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #59, March 1999

Readers sound off.

Acknowledgement

We would like to thank you very much for publishing (with a wonderful cover) our article “Wireless Networking in Africa” in the December 1998 issue. We have received many compliments and requests for more information. I wish to mention that Dr. Emmanuel Ekuwem's name was omitted in the acknowledgments and request you add his name to that section:

> Our appreciation also goes to Dr. A. Nobile, one of the coordinators of the Programme, and Dr. E. Ekuwem, Mr. A. Maggi and C.E. Onime who are actively participating in this Project.

This is very important to us as he has done a lot for the project and merits the acknowledgment. Thanks very much.

—Enrique Canessa canessae@ictp.trieste.it

Advertising

Regarding Mr. Havlik's letter in the December 1998 issue complaining about the presence of advertising, it is easy for readers who don't like the ads to skip them. I personally value the ads. I want to know about the commercial products available for the Linux world, and I want to know about the firms that deliver these products. I vote for the ads.

—Chuck Jackson chuck@jacksons.net

qmail Article in June

I was just re-reading Mike Thomas' article “Virtual Domains and qmail” in the June 1998 issue of *Linux Journal* and noticed that the qmail_db_lookup script contains the test:

```
elseif ($db_local_address eq $recipient) {
```

which could fail if the values of the two variables do not have the same case. This could happen, since e-mail addresses are case insensitive.

Assuming all the entries in the database table are lower case, I would do earlier on:

```
$recipient = lc @_[scalar(@_)-1];
```

or something equivalent. Of course, defensive programming would require that we not make the assumption and change the test to:

```
elseif ("\L$db_local_address" eq $recipient) {
```

so one can sleep easy at night.

—Gyepi Sam gyepi@praxis-sw.com

Xi Graphics Salute

I am writing to salute the support team at Xi Graphics. Never in my computing career have I worked with a more prompt or helpful support staff. Their response time is incredible; I submitted a Support Request to their web site and got a response in five minutes! Via e-mail, Bryan (the Xi Graphics support engineer) and I worked on a weird refresh rate problem I was having with Laptop Accelerated X. It took a few rounds of e-mail back and forth, but we got the problem straightened out and Bryan was patient and helpful through it all.

Three cheers to Xi Graphics support!

—John Duksta jduksta@bbn.com

Linux for Macintosh

I am disappointed that the first substantive mention of Linux for the Macintosh (“Linux for Macintosh 68K Port” by Alan Cox, January 99) in the several months I have subscribed is an aggressively hostile article about several generations of old Macintosh hardware. Sure, people who know anything about Macintoshes will also know the current hardware and Linux picture for Macintoshes, but a very significant fraction, likely a large majority of the readers, will read the

article as being less about the task of porting Linux (which it is about) and more about how inferior the Macintosh is.

There is a vigorous Macintosh Linux community, both an official Apple version (http://www.mklinux.apple.com/) and the even more popular LinuxPPC (http://www.linuxppc.org/). Current Apple hardware with Linux is as good as any. The author's second sentence, "... Apple does not want other operating systems on its machines" is refuted by the aforementioned official Apple Linux.

I would say more, but I am probably going to get relegated to the legion of Macintosh zealots anyway. Believe it or not, I don't want Apple or Wintel to take over the world.

—Jeffrey L. Wragg wraggj@cofc.edu

Alan did mention the many users of Linux on the Macintosh, and as you point out, his article was meant only as a report on how to port the kernel, not how to use Linux on the Macintosh. We have had articles in the past on using Linux on the Macintosh (see issues 19, 31, 37 and 45) —Editor

Red Hat Phenomenon

I would like to join the chorus of voices of frustration with the "Red Hat" phenomenon. If you follow other UNIX journals, you will see a growing annoyance amongst UNIX users with Red Hat Linux. There does not seem to be any reasonable explanation anywhere for the strange Red Hat implementations. To quote Mike Borowiec on page 9 of the December 1998 *Performance Computing Magazine*: "I've since wiped the stain of [Red Hat] Linux from my machine ..."

Mike's problems are the same as everyone else's in the UNIX community. Red Hat is unconventional in layout, difficult to install, extremely difficult to reconfigure and deficient in basic tools. The worst problem is that Red Hat requires extensive editing of C source code and rebuilding of the kernel. This is not a Linux problem—it is a Red Hat problem (other Linux distributions follow conventional UNIX methods). What is Red Hat's objective in making and promoting such a difficult implementation? Where are we heading if Red Hat is creating such a negative reaction against Linux among professionals? None of it makes even the slightest bit of sense to a UNIX user.

The question you need to answer for us (since we cannot figure it out on our own) is: why is Red Hat doing this to the Linux community? And why are so many industry and media types (including your magazine) getting behind Red Hat? Assuming that what Red Hat is doing makes sense to somebody, we

readers really ought to have some kind of explanation so we can understand and evaluate what's going on.

—Reilly Burke reilly@aerotraining.com

Sorry, you will have to ask Red Hat about their policies. I am not in their confidence. However, Red Hat does seem to be the most popular distribution available, so they must be doing something right —Editor

Porting MS-DOS Graphics Applications

Thanks! Jawed Karimi's September 1998 *LJ* article (“Porting MS-DOS Graphics Applications”) on basic graphics got me going in the right direction with SVGALIB. (I am a Linux newcomer and had never heard of it before then.) Sometimes, short and sweet articles are the best.

—John E. McGovney compgraf@bellatlantic.net

Samba's Encrypted Password Article

I'd like to mention that I disagree with author John Blair regarding his article “Samba's Encrypted Password Support” in the December 1998 issue.

In the last paragraph, he states “Finally, to allow users to update their encrypted password, set the permissions on smbpasswd to be setuid root ..”.

This should not be necessary, since **smbpasswd** acts like an SMB client, asking over an SMB session to change a user's password. This is the reason you should include in smb.conf the “allow hosts = 127” line, or leave it blank.

I have also used **pam_smb**, which can be found on the main Samba FTP site, so that other services can authenticate with Samba. On my system, my users all have their passwords in /etc/shadow “starred” (*).

About the magazine, it's great!

—Celso Kopp Webber webber@sj.univali.rct-sc.br

Symbios Controllers and HP Scanjets

The “free” controller shipped with a lot of scanners is the Symbios 53C400A. It is not, as the writer of the reply (“Best of Technical Support”, January 1999) suggested, 53c8xx compatible but is instead a 5380 variant.

It is supported by current Linux 2.1.x series kernels, although its lack of an IRQ and 8-bit ISAness make it a good candidate for the bin, not a multiuser OS.

There is an equivalent patch for 2.0.x in the SANE mail archive.

—Alan Cox alan@terrorserver.swansea.linux.org.uk

Linux Journal Enterprise Solutions

I think the supplement is a very good idea-I very much enjoyed this special issue.

I would like the addition of a Table of Contents as in the regular issues. This would be helpful because I (and perhaps others) photocopy the Tables of Contents and put them in a three-ring binder, where they are easier to scan than the issues themselves.

Thanks for an excellent publication.

—Ray Liere lierer@cs.orst.edu

One of the weird rules of the Post Office is, if it has a table of contents, it cannot be called a supplement. Go figure —Editor

Csound

I just wanted to say “Thank you” to all at *LJ* who did such a great job presenting my article in February. The magazine keeps getting better and better.

By the way, my bio states that I maintain the “official” Linux Csound, but that is no longer true. As of late 1998 I have worked mainly with the development version available from Nicola Bernardini's site at AIMI. The official version is currently maintained at Bath by John Fitch.

Best of the New Year to all of you!

—Dave Phillips dlphilp@bright.net

Linux and Spouses

For years I've maintained a dual boot machine at home with both Windows 95 and Linux because my spouse didn't feel she could learn the intricacies of Linux. So, it was a great pleasure to see KDE chosen in your 1998 Reader's Choice poll. It was with KDE and StarOffice that I finally won her over to the good side.

The ease of a standard desktop which was more intuitive than Windows 9x has her singing its praises. I thought I would never see the day. It should be pointed out that a big turning point for her was the day she accidentally started three different StarOffice sessions and was amazed that the system didn't crash. It did take awhile for those monsters to load, but once they finished, she was back in business. Windows 9x would have died at this point.

Now all I need is to find a calendar program which comes close to Calendar Creator Plus and Windows 9x will just be a bad memory in our house. Maybe korganizer will fill the bill when it's completed.

—Randy Kyrk rkyrk@anet-chi.com

Archive Index Issue Table of Contents

Advanced search

More Letters to the Editor

These letters were not printed in the magazine and appear here unedited.

Virtual Services

Last month, my friend asked me on how to provide virtual services (web, ftp etc), so I scan through my old collection of *Linux Journal* looking for solution. When I look at Issue 35 (March 97), there is an article “A Guide to Virtual Services”. I eagerly read the article. The article provide the information that I was looking for, but in that issue only describe for www service saying that it will continue the month after. So I immediately open up issue 36, but to my disappointment, the article was not there. I scan through few Issues after that, but there was no luck. Although this is an old Issue, but I think *LJ* should not let this kind of problem happen again. I remember in one of the Issue on TOC, *LJ* advertise an article up coming, but I could not find the article in the month after.

Overall, I find *LJ* to be very useful and informative, but there are few little things that can be improved. Keep up the good work.

—Meor, Malaysia, ridzuy@petronas.com.my

---

please improve table of contents with html links

LJ is a very great journal and I enjoy to read it every month.

but:
1) It will be great if you put the Internet links (http, ftp...) referenced in articles with the table contents available on the web.

2) I'd like to see a more powerful endorsement to the open source model. Linux couldn't exist and live without it, even if non open source program are welcome. For example you could systematically mention the status of projects: open source/closed source and if open source the type of license (GPL, LGPL, BSD..)

And with tutorial and other documentation you could promote the open content license.

Happy New Year and thanks for the high quality of *LJ*.

—Jean-Marie Renard, renard@univ-lille2.fr

Nope, Adm. Hopper did NOT coin "bug"

No doubt there's some urban legend site somewhere that debunks this myth. I believe Edison talked of "debugging" projects, and that Adm. Hopper herself tried to correct the myth every chance she could.

—Felix Morley Finch, felix@crowfix.com

A New Beginning

Over these Christmas holidays I have found time to try and install a particular, well-known, commercial version of the latest Linux kernel which I acquired some weeks ago. It has not been an easy task as, for one thing, I found the handbook very confusing, as I am a relative, if not intermittent, and maybe incompetent newcomer to Linux. Although, I have taken *LJ* for nearly four years now, and I did attempt to install an earlier kernel last year, but without success and I gave up; having spent many frustrating and wasted hours. In fact, thinking about it, I have over these past four years attempted on a number of occasions to get various 'makes' of Linux running on my machine(s), but I have never been successful beyond the basic installation and an occasional functioning X server. Abundant time and patience I do not have. But the thought of an alternative to MS Windows still appealed. I had hoped that through the suppliers much sung set-up tool I would at least have a fully operational Linux system running on my PC and, although I have now installed it and the X server, with the KDE desktop running, I find that I now cannot install my printer, nor can I connect to my ISP for e-mail and WWW. So I cannot use it, even though it appears to be running as a basic system. Eventually, of course, I would have liked to install various applications to help me in my work and life generally, but I fear this will be just another courtship without a marriage.

I have looked through the various, and apparently applicable HOWTO documents, from the usual web sites (using Microsoft Windows 98 and Internet Explorer, which were very easy to install) but, again, I find them very confusing, extremely time consuming and very uninteresting, as they appear not to have the solution I am seeking to my problems. No doubt technically brilliant - but useless to a non-techie!

I have even asked questions on the relevant Usenet Newsgroups, but no-one gives the solutions, they just keep asking for help themselves! I have reached the conclusion that Linux is really an amateur hobby software for technically advanced, masochistic computing academics, who enjoy making life difficult for themselves - but, regretfully, also for others, who are caught in their web by smart promotional material and feelings of inadequacy subliminally poured upon them, given false hopes and rapidly leading to unfulfilled ambitions.

I always read in *Linux Journal* about how wonderful Linux is (understandably so), and how Microsoft is a dinosaur - holding back the horizons of computing and making mediocre use of today's technology. Every issue of *LJ* tells us how many software companies are now writing for Linux. And, there is even an Enterprise supplement to the latest issue

of *LJ* attempting to instill even more confidence in all those people out there (like me) who would love to be released from the bonds of Bill Gates. But, alas, I fear you are all chasing the rainbow!

Until you make the installation of Linux and its application software as easy to install as Microsoft, and others writing software to run on Windows 9x, you will never be taken as a serious player in the PC OS game. As much as I, and no doubt many millions of others like me, would wish for Linux to carve a large slice out of Microsoft's cake. No, until the ethos of Linux changes it will remain a Sunday afternoon game for most of us.

But, perhaps, that is what Linus Torvalds wanted Linux to be. A mere diversion, a game on a rainy day for millions of computer users who are continually frustrated by the hang-ups, glitches, idiosyncrasies and poor performance of MS Windows. A subversive, flanking attack on the mediocrity we have all now come to accept. A sniper, a terrorist, perhaps. Without wishing to win any war but just, simply making a point! Making us believe there is something out there which is better.

The foundations to a new religion, maybe. The end of that rainbow, perhaps? But I, and I suspect many others, are not convinced Linux has any answers, let alone the solution. We need the messiah now, to interface the maker(s) with reality. To interpret the commandments for us. A few disciples to spread the faith to simple-living, working people who are just too busy - or just not interested - in how the 'world' was made. We now want to exploit it for our own gains and evolve as good Linuxians; putting the 'darkness' to task and hopefully, eventually behind us.

Is there no life after Microsoft?

—“Roger Reeves”, roger.reeves@dial.pipex.com

Beginning users

Linux is now getting to the point where DOS was in the early 1980s. The more users that experiment with it at home and learn it's benefits the faster Linux will be appreciated by the average and beginning user. That user is looking for a stable system he can work with and understand. Windows doesn't even come close to this requirement. These beginning and intermediate users should be at least part of the focus of your magazine. If you take the fear out of people using and modifying their computers Linux's use will soar through the roof.

—James L. Caudle, jcaudle@pdq.net

Caldera Open Linux Base 1.2

I have noticed that your magazine supports Caldera products. I am curious to know if you have had the experience of Open Linux developing problems as the OS was used over a period of time.

I purchased my copy of Open Linux in May of 1998. My copy is also registered. Now that I have used Linux and am beginning to understand how it works, I find that things in Open Linux have quit working. I have been careful not to alter any script files (I am not yet proficient at C\C++ programming). At first some of the desktop links did not work. Later, the menu choices in the Desktop box quit working. Recently, Netscape fails to access the Internet. The help files and Star Office have never worked at all.

I am sure that for those of you that are well experienced with Linux that these are minor and trivial problems to fix. Unfortunately for me, I am a newbie. I have found that Caldera does not offer any guidance or help on their web site. Contacting Caldera and paying for the support they could offer is not financially possible. For such a great, new, and an alternative to Microsoft, Open Linux has been a disappointing and frustrating experience. Thanks for your time.

Rick Sams, rsams@Hocking.net

Re: BTS Shutting Down

In the Jan. 1999 Best of Technical Support in the *Linux Journal*, Thomas Okon asked about a utility to allow a user to shutdown the system without having to login as root. The solution is at sunsite as ftp://sunsite.unc.edu/pub/Linux/system/admin/su/usershutdown-1.1.tar.gz

—Andy Holder, aholder@bellsouth.net

Letter to the editor: Red Hat beware

I'm running Caldera 1.3, but I was curious about the Gnome desktop so I ordered the Red Hat power tools package and the freeware games that go with it. I figured Caldera supports rpm's so there would be no problem.

Well, there's a problem. The rpm's won't build due to failed dependencies. I downloaded what it said I needed and installed it. I still haven't got it to work. Bottom line: Instead of playing with a cool desktop and the games that go with the package I'm pissing up a Red Hat rope.

Selling free software that only runs on a proprietary OS is no different than what Microsoft and flavors of UNIX have done for years. It will take a lot of unforseeable good will on Red Hat's part before this user considers them a viable option again under any circumstances.

—Bob Lazarski, blazarski@anatel.com

BTS

Regarding the answer to “Updating Web Site” in Jan 99 issue.

By reading the man page, and config files of mirror I understand the update goes FROM remote TO local site and this is not what Grim_Sweeper@softhome.net wanted (If I am not wrong! ;))

The only two programs I found (I haven't tested them yet) are:

http://www.lyra.org/sitecopy/
triphop.dyn.ml.org

Best regards,

—Rafael Cordones Marcos, rafacm@bcnartdirecte.com

Your publication

I am a relatively newcomer to Linux, and I recently started my subscription to *Linux Journal*. Although I find the *Journal* useful, the level of the articles are way over my head. It is my impression that, now that Linux is expanding its horizons and developing a much larger audience, the need for more basic articles becomes a very important issue. The idea that Linux is for hackers is obviously outdated, but this needs to be reflected in your publication. Thank you for your attention,

Helio J. Malinverni, M.D., hmalinv@cybercomm.net

BTS: User shutdown

Issue #57 column you didn't know of a better way to enable user shutdowns. On my 3 user system, where I don't worry overly much about security concerns, I solved that problem by set the suid bit on the halt and reboot programs. On more security conscious systems one could write a suid script file that checked users presence in /etc/shutdown.allow before executing the real halt or reboot program.

It does seem rather odd that the shutdown command doesn't do this itself doesn't it.

—Kenneth Corbin, kenc@pioneer.net

Shuting Down

I was reading the Best of technical support (January edition) yesterday when I read two things that might have other solutions.

First I would like to remark that with the Atapi Zip drive the preformated partition is /dev/hdX4 and not /dev/sdX4. It is of course an IDE device.

Then the second more important thing is the question about shutting down. Thomas Okon wanted to let another user shutdown the system. You wrote the only possible the three finger salute. However there is another possibility .. it's sudo , sudo allows you to give certain users the rights to use specific commands with root permissions. Probably there

are other tools that do the same but sudo is the tool that I've been using to give my users the root permissions to certain commands.

It might be useful to some other question in the future

—Kris Buytaert, Kris.Buytaert@advalvas.be

Enterprise solutions

Like everybody else I try to promote Linux as much as possible and think a lot about how to use Linux in business environments. So I think it's a great idea to bring out the *Linux Journal* supplement: enterprise solutions!

I've red a story of a guy trying to implement Linux and StarOffice in a small office and made mistakes that had nothing to do with Linux but is good to warn the community about so here's the URL:

users.smileys.net/~leonb/staroffice-wars.html

Thanks a lot for the great work you are doing, *Linux Journal* brings me a lot of fine reading hours !

greetings

—Ben van Scheppingen, besc@dds.nl

BTS

Re: Best of Technical Support - Shutting Down, *Linux Journal*, January 1999, Page 61

Thomas Okon (okon@math.tu-dresden.de) wrote concerning how to allow a non-root user to shut down the system. I offer a solution that worked for me with Red Hat 4.2, and now with Red Hat 5.2.

I set up a user that could shut down my Linux machine when I logged on as that user. The only purpose that user had was to shut down the machine.

I manually performed the following steps to enable this capability. I always saved a backup copy of the original files before I made any changes.

- Edit the “/etc/passwd” file. Add the new user (I called mine “down”), giving him the same UID, GID, user info, and shell fields as the root entry has. Example:

```
root:<passwdfield>:0:0:root:/root:/bin/bash
down:<passwdfield>:0:0:root:/home/down:/bin/bash
```

(A password for down is optional. Allows a user to simply enter “down” at the login prompt to shut the machine down, or provides additional security to prevent unauthorized shut downs.)

- Edit the “/etc/group” file. Add the new user as an additional user for groups root and users, and add an entry for the new user. Example:

```
root::0:root,down
users::100:bob,down
down::500:down
```

- Make a new “/home/*new_user*” directory, set the permissions to 755 (rwxr-xr-x), set the owner to down. Example: (`ls -al report`)

```
drwxr-xr-x  500  down   1024  date  down/
```

- Copy the .bashrc and .bash_profile from another existing user's home directory to the new user's home directory. These should be the original, unaltered files.
- Edit the new user's .bashrc file, and add to the bottom of the file “/root/down” (without quotes.) (Save a copy first.)
- In the “/root” directory, create a file called “down”, and enter the following:

```
echo "sync"
sync
echo "sync"
sync
echo "shutdown -h now"
/sbin/shutdown -h now
```

  Set the permissions of the “down” file to 777 (rwxrwxrwx), and set the owner and group to root. You can probably use any shutdown options you wish. I chose to use the above.
- I am not sure if this is really necessary, but I did it anyway. Set the root directory's permissions to 755 (rwxr-xr-x).

When I want to shut the machine down, I log off the machine, and then simply enter “down” at the login prompt. The down user's “.bash_profile” file is executed, followed by the “.bashrc” file, which executes the “/root/down” file. The echo lines are displayed, and the machine is gracefully shut down.

—Bob Van Meter, bobvm@rocketmail.com

Please consider

another issue of database articles. I know Feb 1998 was but much has happened. I have all of the ports (Sybase, Oracle ...) and would love to read information from experts on things as how to connect clients to a server (no real examples) ...

love the magazine.

—Dale Smyth, dales@emeraldnet.net

> *Our November 1999 issue will again focus on databases and we have a 3-part Sybase tutorial coming up soon. —Editor*

QuarkExpress Port

I use Linux everyday. I say this simply to let you know I'm not an M$Clone. However, I also use Win95 OSR2 at home, and NT4.0 at my client site (CertCo).

I want to let you know that Lydia's problem is Quark's sole responsibility. Their Win95 port is rotten. A couple of contracts back the company gave the new Marketing Comms lady permission to use Quark. Since it was a Wintel shop, she purchased the Win version. And as you know, it crashes. A lot. I worked with her for about three weeks trying various combinations of HW/memory/installed drivers. We never found a combination that was satisfactory.

She eventually gave up on it and began using a combination of FrameMaker and M$ Word. I really doubt that combo is suited for your tasks, but it worked for brochures, ads, and collateral.

I'm not defending M$ by any means. I just wanted you to know that the problem is squarely in Quark's lap. They'll deny it if you bring them to task, saying that Windows is to blame. But they're quite wrong. I run PS 4/5 as well as AI 7/8 on my Win95 box and have yet to crash it even when doing complex layouts. So Adobe is either heavily into black magic or they have better coders. Personally, I think it's the coders.

Happy New Year!

—Mike Adams, AdamsM@certco.com

"Resources" as part of on-line resources!

There usually is a “resources” section at the end of articles which contains references and URLs.

Why are not those included on-line? If they were, readers would save time by not typing the URLs!

—Farid Hamjavar, hamjavar@unm.edu

> *Just one more thing for me to remember to do. :-) I'll put it on the request list but no promises. — Editor*

Editorial Suggestion

Through your publication, you could provide some services that would foster Linux acceptance into the main stream by making it more approachable. Please consider these two suggestions:

SUGGESTION #1: Demystify the fabric of “UNIX tradition” that is part of understanding most of what happens during the use and administration of a Linux box. If you don't know these traditions, you are often lost without a clue.

SUGGESTION #2: Add a survey page to your web site collecting hardware in actual use and the applications in use on Linux boxes. Differentiate between hobby/residential, educational/student, and commercial installs. [Do something to sort out survey stuffing web crawlers (sigh)] and publish the results as monthly histograms.

DISCUSSION #1: Let me illustrate this fabric by example. Years ago I got a “real AT&T UNIX” system and its wall of paper documentation. This was my first exposure to __UNIX__. I opened the books and was instantly lost in a printing of what I learned were “man pages” bound in alphabetical order. Then someone told me to look under 'G' for “Getting Started”.

Many of the Linux newbies that we want to attract are windows power users. They have a tradition that says, “use documentation as a last resort.” The Linux tradition relies heavily on man-pages, HOWTO, FAQ and other online docs. When you couple this chasm of difference with the “one program one function” philosophy, a Linux user will often need a road map to the several different sets of docs required to learn how to deploy some feature of interest. *Linux Journal* could publish these road maps.

I know that you've seen books or training materials that say, “... if you want to do XXXX, then read the following chapters in this order...” Sort this out and publish it for the newbies (and not so new bees). For example, “If you want a Linux box that does most of what the typical home office Win9x box does, then AAA, ..., ZZZ” Another example, “If you want a Linux box for a SOHO firewall/gateway, then AAA, ..., ZZZ” These could begin as web pages of document links on the *LJ* site , evolve into multi-part *LJ* articles and some eventually become _handbooks_ under the SSC banner.

DISCUSSION #2: If a person, SOHO or corporation wants to “try out Linux” and wants minimum risk, there is some homework to discover a set of compatible hardware, a distribution, and required errata or patches that will work. I found most of what I needed on USENET, but was quite suspect of the results. I wished for a moderated source of “this collection seems to work” information. You could be that source.

I've been a “computer geek” since 1969, and I consider myself both knowledgeable and experienced. In spite of professional experience with both “real AT&T Unix” and with “real UCB UNIX”, I often find myself lost in confusion over the rationale behind many tools and tasks of system use and administration.

—Daniel M. St.Andre', grillon@nuthena.com

LyX Article Errors

When I received my January, 1999 issue of the *Linux Journal*, I immediately noticed the article about LyX. I then proceeded to read this article. I was, unfortunately, struck by numerous inaccuracies.

First, I was dismayed—nay, insulted—to see the statement: “Now back to the documentation. [\ldots{}] most of the files are not too detailed [\ldots{}].” Since I am the former editor of the LyX Documentation Project and personally wrote about half of the User's Guide, I know for a fact that this statement is flat-out false. Seriously, though, the article's description of using a file descriptor to open a manual named Main.lyx does not describe version 0.12, but one of the earlier versions of the 0.10 release. The file descriptor long ago disappeared, replaced by 6 menu entries for each of the primary manuals. These manuals, in turn, are highly detailed and complete. I highly recommend Amir Karger's Tutorial as a place to start. Only the Customization and Reference manuals remain incomplete and crufty.

A second error is the description of an error message due to a missing X-server font. I am perplexed; I never saw any such bug report cross the LyX developers' mailing list since the 0.9 development series over 3 years ago. Back then, Asger Alstrup Nielsen worked hard to improve the font-drawing code. His continued work over the past few years make loading even a huge document [e. g. the User's Guide] quite speedy. Only a 486 [such as my laptop] will show any delay due to X-server font loading.

In another mistake, the article states: “Change the template file by hand using any plain text editor. Please, please, please do not edit any LyX file by hand with a text editor. You will render it unreadable by LyX [unless you really, really know what you're doing, and even then, you'll screw up occasionally—I speak from experience].

There are several minor mistakes, including:

- There is no longer a ~/.lyxrc file. Instead, there is a ~/.lyx/ directory where you can put all sorts of user customizations.
- While you can compile LyX with version 0.81 of XForms, the LyX Team strongly recommends use of version 0.88, which is far less buggy. LyX will also run much better.
- LyX is not a LaTeX editor; it's a “Document Processor” [to quote the 6 manuals, the README file, and the man page]. You don't need LaTeX to use LyX—I'm running LyX right now on my old laptop, which doesn't have LaTeX installed. LaTeX is merely one of the ways LyX generates printable output. Another way is via the LinuxDoc/SGML Tools.
- LyX is not WYSIWYG, and never will be. It is WYSIWYM [==What You See Is What You Mean], a term defined in the 6 manuals, the README file, the man page, as well as in e-mail messages on both mailing lists.

LyX doesn't waste time, code, and memory trying to reproduce the printed page, a superfluous feat. Instead, it provides you with contextual, visual “cues” that imply the printed result in a natural, often intuitive fashion.

I don't intend to belittle Ulrich Quill's efforts in writing this article. I only wish he had made certain his reporting was accurate. Had he contacted the LyX developers' mailing list, we could have assisted him with the article, as well as told him about certain important changes:

- The new official LyX home page is at http://www.lyx.org/. It contains links to everything else, including a more up-to-date link for KLyX.
- The official LyX FTP site is ftp.lyx.org [yes, it's still at via.ecp.fr, but should that ever change, ftp.lyx.org will point to the new location].
- By the beginning of January, 1999, LyX version 1.0.0 will be out. It will include complete manuals, more supported document classes, integrated LaTeX input, and, of course, many bug fixes.
- For version 1.1.0, LyX and KLyX will converge due to the Team's efforts towards GUI-Toolkit independence. If you're on a Toolkit Jihad, it will be easy to port LyX to The One True Toolkit. [We'll initially support XForms, QT, and GTK.]

I have one final note to anyone wishing to write an article for the *Linux Journal* and get their name in lights. When writing about a program, please do contact that software's development team! I often see articles about Linux in the professional IT journals at work, articles that are full of falsehoods and other bad reporting. [One such article stated that the many Linux distributions were binary incompatible. Heh?!] If we can't report about ourselves with accuracy, how can we expect others to do the same?

P. S.: These opinions are my own, not those of the LyX Team as a whole.

—John Weiss, jpweiss@idsi.net

---

happy hacking keyboard

I just read the review of the Happy Hacking Keyboard in *LJ*, and I couldn't help but notice that it looked very familiar to the keyboard I got at Fry's electronics for all of $20. Granted, the control and alt keys were not in new places like the HHK, and it did have that darn windows key, but I'm willing to put up with it (:

—Joshua Colvin, jco@dre.cse.nau.edu

Re: Linux Technical Question

This is a response to a question in the January '99 issue about mounting a Windows 98 Zip disk. The response indicated that the partition is actually partition 4 but that Peter did not know why. Here is the answer: Partition 4 corresponds to the DATA partition on a Macintosh drive. This allows the Zip/Jaz drive to operate on both PCs and Macs (This was

gathered by way of my usage of a Jaz disk which has the same consideration).

—Daniel Davis, elminstr@bellsouth.net

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #59, March 1999

Our experts answer your technical questions.

Modem Compatibility

Can I use an internal PCI modem with Linux? I have been looking all over the web and have found no documents about this. —Luis David Cardenas, luisdav@liza.net.mx

The answer will depend upon which kind of modem you have (Plug-n-Play, winmodem etc.). If your modem has these switches (also called jumpers), configure it to use an IRQ/Com (IRQ3, Com 2 for example) and simply create a link (as root) to it:

```
cd /dev
ls -s /dev/cuaX /dev/modem
```

where the X is directly related to the Com port you have chosen (Com1 = cua0, Com2 = cua1 and so on). If your modem is Plug-n-Play, you will have to create an /etc/isapnp.conf file and configure it accordingly. —Mario Bittencourt, mneto@buriti.com.br

Internet Connection Broken

I recently upgraded from Red Hat 5.1 to 5.2 and was surprised to see that my Internet connection no longer works. I set up my PPP connection through Network Configurator. I have no problem dialing and logging in, but when I pinged one of the sites, e.g., www.idl.net.au, I got this message:

```
ping: unknown host www.idl.net.au
```

Which is strange, because it used to work under 5.1. —Genesis Elliott, genesis@mail.idl.net.au

I suppose your ping is just a test and other things that used to work also stopped working (I am saying that because DNS might not be working, or the host you are pinging could have disappeared—I have seen that happen). Let's suppose your problem is truly linked to the upgrade. Make sure that /etc/resolv.conf still has correct values **(netstat -nr; ifconfig)**, or else make sure you cannot ping hosts by their IP addresses (rather than host name). Then, if that fails, under interfaces/ppp/communication check the debug box, and do a **tail -f /var/log/messages**. Bring the connection up, and the messages from syslog in /var/log/messages should give you a clue as to what is wrong. —Marc Merlin marc@merlins.org

Chinese Big-5 Display

How do I display a Chinese charactered document under xterm and web browser? —Chi-Chih Chen, chchen@csci.csusb.edu

You can use **cxterm** to display Kanji under Linux, and it offers several ways to input Chinese. Netscape can also be configured to display Mandarin in the View/Encoding menu, both simplified Big5 encoded Kanji and traditional Kanji as used in Taiwan.

You should read the Chinese Linux HOWTO at www.phys.ntu.edu.tw/~cwhuang/pub/trans/linux-howto/Chinese-HOWTO-3.html (it has a link to cxterm). —Marc Merlin, marc@merlins.org

Options at Compile Time

I know how to compile a new kernel and pick support for various options. Is there a way to do a sort of reverse lookup to find out which options were picked at compile time? This is needed in order to determine which options were compiled into a particular kernel I have floating around but for which I don't have the accompanying config file. —Ian Gilchrist, ian.gilchrist@mail.com

This information is not saved as any type of text data specifically, but if you are creative you can find other ways to get this information. Each time Linux is compiled, a map file is also created. It contains the function names in the kernel, and with some work, you can probably figure out from there what you need to know.

It is a good idea to save the .config file with each compiled kernel. It has all the information you need; I often compress it with **gzip** and save it in the directory where I store my kernels. This is much handier than you may think. There is nothing worse than trying to compile a kernel for a two-year-old system with a really old LAN or SCSI card, when you cannot remember how you got it to work the first time. —Chad Robinson, chadr@brt.com

Accessing Home Pages via FTP

We are a small ISP and have used Windows NT, WebTEN and MachTEN for our services. We have now changed to Linux.

We have set up Linux, and we want our customers to be able to use ftp to put their home pages on the server. This works fine, but they also are able to go up the tree to see the /home/dir, and even more, the whole hard drive. I want to change it so that they can go into their /home/$USER and no further. I have tried to use the **guestgroup** option in /etc/ftpaccess to no avail.

How do I do it? What is the proper syntax? Can anyone send me a working line out of the /etc/ftpaccess and the /etc/passwd file? —A.H.J. Mittendorff, gr0upe@technologist.com

The easier way is to create an FTP group (let's call it ftponly) in /etc/group and change your creation script to add the users with this group. Then edit your /etc/ftpaccess and add **guestgroup ftponly**. Now, if you try to log on using ftp, you should be able to see only your directory. Note that since FTP will **chroot** to the user's directory, you will also have to add a /bin directory with a statically linked **ls** version.

Also, check that your /etc/ined.conf has a line like this:

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
```

—Mario Bittencourt, mneto@buriti.com.br

Setting Up kermit

I have searched through all of the suggested resources and have been unable to find an answer. How can I set up **kermit** or Linux so that they can talk to each other? I am trying to use ckermit 6.0.192 to dial a pager, and I get the message "Warning: terminal type unknown ..." each time I start up kermit.

Once I get to kermit and try to dial a number, I get the error message "unable to initialize modem". I have used **minicom** to dial my modem and it works fine, but the application (Big Brother) which uses the modem is set up to use kermit. —John Willoughby, jwilloug@vvtv.com

Try to force the TERM variable to vt100 before starting kermit:

```
export TERM=vt100
```

At the kermit prompt, try this sequence (I am assuming /dev/modem is a correct link to your modem device):

```
set line /dev/modem
set speed 57600
connect
atdt...
```

This works fine for me. —Pierre Ficheux, pficheux@com1.fr

Partitioning

I am a newbie at installing Linux so I am not quite sure what I may be doing wrong. Here is my problem:

I am trying to install Red Hat on my machine, which currently has Windows 95 on it. I have a 6.4GB hard drive and I have used Partition Magic to free up about 2GB of space. Windows 95 is on three partitions; one of those is an extended partition. I decided to use Disk Druid to make the Linux partitions because I am not totally familiar with **fdisk**. I first made a root partition, which worked fine, but if I try to create any other partitions, I get an error with unallocated partitions and the reason is “no free primary”.

I don't really want to format my hard drive, but if it comes to that, so be it. —Ben Barth, bebart@dave-world.net

You may only have four primary partitions on your hard drive. If you need more, you can make an extended partition instead of a primary. Under DOS, this is how logical drives are created. Linux has no problem residing on an extended partition, so you should be able to install it once you create the extended partition. Remember that you will probably want to make two: one for the file system and one for the swap space (usually about twice your system RAM in size). —Chad Robinson, chadr@brt.com

Getting Started with Applications

How do I run a new application or install it into X? For instance, my Red Hat bundle had a CD with lots of software on it. I used RPM to install Word Perfect 7.0. Now I have an /opt directory with a /wp70 directory under it and a bunch of subdirectories under it. Which one is the executable? What magic word or process am I missing to make WP run? All the files seem to be there. This is the first question I haven't figured out, and I have spent a lot of time trying. My system is working great but this is just too nebulous for me. —Sean Wyatt, wyattcfi@compfuture.com

Usually, applications that install in /opt install a link to their binary in /opt/bin. So, all you should need to do is put /opt/bin in your PATH. In the /etc/profile file, you can add:

```
PATH=/opt/bin:$PATH
export PATH
```

Should the executable be absent from /opt/bin, you can locate it using:

```
find /opt/bin -follow -type f -perm -700
```

then create a link yourself and add whichever directory it lies in, to your PATH. —Marc Merlin, marc@merlins.org

Mysterious Messages

After recompiling the kernel, I get a message along the lines of:

```
modprobe: cannot find net-pf-4
modprobe: cannot find net-pf-5
```

at various times. It does not seem to affect the behavior of the system, but I would like to know what it means and what I should do about it. —Frank McCabe, fgm@fla.fujitsu.com

net-pf-4 and **net-pf-5** refer to support for IPX and AppleTalk, respectively. You can avoid the modprobe messages from appearing at boot time by including aliases for each in /etc/modules.conf (or /etc/conf.modules), i.e., add these lines to the file if they are not already there:

```
alias net-pf-4 off
alias net-pf-5 off
```

Most of this information can be found in /usr/src/linux/Documentation and I would recommend becoming familiar with the resources in this directory. Note: /usr/src/linux is just a symbolic link to your current source tree of the Linux kernel. —Andy Bradford, andyb@caldera.com

Archive Index Issue Table of Contents

Advanced search

Partners—Pacific HiTech and Panasonic

Marjorie Richardson

Issue #59, March 1999

This bundling deal is believed to be the first time for a major PC manufacturer to offer Linux preinstalled on desktop computers.

In looking for international news for this column, I found one spurious rumor that Hitachi would be using Linux and one true rumor that Pacific HiTech and Panasonic were becoming partners. This is good news indeed for the Linux community in general and for the Japanese community in particular. Our cover is a Panasonic computer running Pacific HiTech's Linux in Japanese. The following information was provided by Lonn Johnston and Craig Oda of Pacific HiTech.

Panasonic, one of the world's largest consumer electronic and computer vendors, will be installing Pacific HiTech's TurboLinux operating system on high-end Panasonic desktop computers. This bundling deal is believed to be the first time for a major PC manufacturer to offer Linux preinstalled on desktop computers.

Pacific HiTech (PHT) is the largest Linux company in Japan with more than 50 percent market share. In December 1998, PHT Linux products outsold Microsoft's Windows NT in Japan, according to Japan's Computer News, an industry analyst firm.

In March, PHT will announce two high-performance Linux server products for the United States market for corporate Intranet and Internet applications. It will also announce a high-end TurboLinux product that will push Linux to new levels of mission-critical deployment. PHT will back its products in the United States with national support from one of the world's largest corporate call centers and training companies. It already has this support infrastructure well established in Japan.

Panasonic is already offering its 275,000 employees the option of ordering their computers preinstalled with TurboLinux and has quietly been shipping TurboLinux to corporate customers on Panasonic computers since December 1998. Pacific HiTech will be the exclusive supplier of the Linux operating system to Panasonic.

The TurboLinux option will be offered on Panasonic desktop computer models using Intel Pentium processors at 400 to 450MHz. While the computers will be sold under the Panasonic brand, the machines will actually be built by Proton under an OEM arrangement. Proton, one of Japan's largest desktop manufacturers, is a subsidiary of Ado Denshi, a high-technology Japanese company. The units will initially be available only in Japan.

The Pacific HiTech and Panasonic deal is a key international development, which will help drive Linux into the corporate mainstream, both in Japan and the United States. Corporate chief information officers have been deploying Linux in selected applications for several years but have been reluctant to embrace the upstart operating system throughout the information technology infrastructure until support and other mission-critical issues have been resolved. By bundling TurboLinux in volume preinstalled desktop machines, Panasonic is addressing many of those issues and bringing credibility to Linux as a viable corporate operating system for the desktop, Internet and Intranet, as well as mission-critical applications.

Pacific HiTech has a long list of software, hardware and distribution partnership agreements with leading companies around the world, including Oracle, Adaptec, Accton, Panasonic, Corel and DDI, Japan's second largest telephone company.

PHT invests significant development resources in its TurboLinux product suite to simplify installation and offer the most stable, robust and powerful Linux operating system on the market. PHT believes customers are better served by Linux as an operating system of choice if they also receive commercial software bundled with Linux. By offering Oracle bundled with TurboLinux, for example, customers gain many benefits: easier installation, performance improvements and one-stop support.

TurboLinux was chosen by Panasonic not just for its technological advantages and Pacific HiTech's support infrastructure, but also for its programming design that easily accommodates languages other than English. TurboLinux is available in three languages: English, Japanese and Chinese (more versions to be announced). While most systems administrators around the world have adopted English as the Lingua Franca of computing, there are major advantages in administering systems using one's native language. For less technical computing audiences, vendors have no choice but to offer local language support. TurboLinux easily accommodates different languages from installation and graphical user interface all the way through to network administration tools. It is a customer-friendly approach consistent with Linux as an international operating system.

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen Dahl

Issue #59, March 1999

ENVI version 3.1, FTLinuxCourse, Alexandria 4.50 and more.

ENVI version 3.1

Research Systems, Inc. announced the release of ENVI (Environment for Visualizing Images) version 3.1, a robust, easy-to-use image processing system that provides analysis and visualization of single band, multi-spectral, hyperspectral and radar remote sensing data. New features include interactive options for working with imagery, more hyperspectral analysis features and support for more data/image/vector formats. ENVI 3.1 is available for Linux and other operating systems. Linux pricing starts at $3,350 US.

Contact: Research Systems, Inc., 4990 Pearl East Circle, Boulder, CO 80301, Phone: 303-786-9900, Fax: 303-786-9909, E-mail: info@rsinc.com, URL: http://www.rsinc.com/.

FTLinuxCourse

Based on Caldera OpenLinux, FTLinuxCourse is a step-by-step installation course written in HTML and available on a CD-ROM in English, Spanish, German, Italian and French. The package includes StarOffice, Communicator and KDE tutorials, complete Linux command references with on-line examples

and more than 500 questions with answers, including 50 tests. The base course costs $59 US. An updated price list is on their web site. The complete version will be available spring 1999.

Contact: Future Technologies, Via Cairoli, 1, 33170 Pordenone (PN), Italy, Phone: +39 434 209 107, Fax: +39 434 209 510, E-mail: sales@futuretg.com, URL: http://www.futuretg.com/FTLinuxCourse/.

Alexandria 4.50

Spectra Logic Corp. announced the availability of version 4.50 of its Alexandria Backup and Archival Librarian software. This new version adds support for the Linux operating system. Alexandria 4.50 has been ported to the Red Hat and Slackware Linux distributions and additional ports are being developed for S.u.S.E., Caldera and TurboLinux. Pricing for Alexandria 4.50 varies according to environment and is quoted on a per-customer basis.

Contact: Spectra Logic Corporation, 1700 North 55 Street, Boulder, CO 80301, Phone: 800-833-1132 or 303-449-6400, Fax: 303-939-8844, E-mail: alexandria@spectralogic.com, URL: http://www.spectralogic.com/.

Applixware for Linux

Applix, Inc. announced the release of Applixware 4.4.1 for Linux running on Compaq's Alpha processor. The package includes Applix Words, Spreadsheets, Graphics, Presents, HTML Author and Applix Data, which provides database connectivity to Oracle, Informix, Sybase and other Linux databases. Applix Builder, a graphical, object-oriented development tool with CORBA connectivity, is also included in the suite. Pricing for the suite is $99 US.

Contact: Applix, Inc., 112 Turnpike Road, Westboro, MA 01581, Phone: 508-870-0300, Fax: 508-366-2278, E-mail: applixinfo@applix.com, URL: http://www.applix.com/.

c-tree Plus V6.8A

FairCom Corporation announced the newest release of the c-tree Plus V6.8A file handler for Linux. This release of FairCom's C ISAM database API offers flexible file limits and enhanced file mirroring. When used with the FairCom Server, c-tree Plus also offers file encryption. c-tree Plus V6.8A is priced at $895 US with full C source code, no royalties, 26+ free development servers and development ODBC drivers.

Contact: FairCom Corporation, 2100 Forum Blvd., Suite C, Columbia, MO 65203, Phone: 573-445-6833, Fax: 573-445-9698, E-mail: faircom@faircom.com, URL: http://www.faircom.com/.

AFS Server and AS Client

IBM's Transarc subsidiary announced its first Enterprise File Systems products for Linux. AFS Server and AS Client are now available for users to add Red Hat Linux to their enterprise environments, enabling interoperability between servers and clients for Microsoft Windows 95/98/NT, Linux and other UNIX operating systems. AFS Server 3.5 and AS Client 3.5 provide a reliable file sharing option with performance enhancements in the file server and the backup system. Pricing for AFS Server begins at $1,995 US; access to the AS Client at $99 US per user. For web-enabled environments, pricing for unlimited users is $6,495 US.

Contact: Transarc, The Gulf Tower, 707 Grant Street, Pittsburgh, PA 15219, Phone: 412-338-4400, Fax: 412-338-6977, E-mail: sales@transarc.com, URLs: http://www.transarc.com/, http://www.software.ibm.com/.

Hamilton

Microstate Corporation announced that its thin-client, cross-platform web server based on Java technology is now available as open-source software, freely accessible over the Internet. Named Hamilton, Microstate's server technology is integrated software that includes most of the infrastructure required by any thin-client application. Companies can easily deploy and maintain large-scale applications over networks regardless of the desktop platform.

Contact: Microstate Corporation, 11166 Main Street, Suite 100, Fairfax, VA 22030, Phone: 800-684-5197, Fax: 703-591-1790, E-mail: info@microstate.com, URL: http://microstate.com/.

Cat5 Reach

Raritan Computer, Inc. announced its new Cat5 Reach, enabling a user to access a PC (CPU) in a clean, secure area from any remote location within 650 feet. KVM signals will transmit over a single, standard install Category 5 UTP cable. Similar to PC-Reach and PC-Reach Plus, Cat5 Reach is a transmitter and receiver pair that lowers the costs of installing and maintaining long-distance

keyboard, monitor and mouse access to stand-alone PCs or multiple computers connected to Raritan's KVM switch products. Suggested retail price is $545 US.

Contact: Raritan Computer, 400 Cottontail Lane, Somerset, NJ 08873, Phone: 800-724-8090, Fax: 732-764-8887, E-mail: sales@raritan.com, URL: http://www.raritan.com/.

TclPro 1.1

Scriptics announced the release of TclPro 1.1 with added support for Linux and SGI platforms. Available for free evaluation download, the evaluation copy is a fully functioning version with a 30-day license key. Support is for Red Hat Linux 5.0 or later on the Intel platform. Scriptics believes the release will work for any distribution that uses the glibc libraries, including the latest Caldera and S.u.S.E. releases. The single-user price for TclPro is $1000 US, or $1200 US when bundled with TclPro Update Service.

Contact: Scriptics Corporation, 2275 East Bayshore Rd., Suite 101, Palo Alto, CA 94303, Phone: 650-843-6900, Fax: 650-843-6909, E-mail: sales@scriptics.com, URL: http://www.scriptics.com/tclpro/.

Wingz Professional 3.0 for Linux

Investment Intelligence Systems Group (IISG) announced the release of Wingz and Wingz Professional version 3.0 for Linux as freeware. The Wingz suite is comprised of three products: Wingz Professional, Wingz and the HyperSheet Runtime. Wingz Professional is a multi-platform, visual programming environment allowing developers to create customized graphical applications for data analysis and presentation. Wide-scale deployment of the standard programs or custom applications bundled with the HyperSheet Runtime are subject to licensing fees. The products can be freely downloaded from http://sunsite.unc.edu/.

Contact: IISC, 8400 W. 110th St., Suite 305, Overland Park, KS 66210, Phone: 913-663-4472, Fax: 913-663-4473, E-mail: us-sales@iisckc.com, URL: http://www.wingz-us.com/.

WordPerfect 8 for Linux

Corel Corporation announced the availability of a multilingual download version of Corel WordPerfect 8 for Linux (English, French, Italian, German, Spanish or Dutch). The free evaluation includes only the application. A shrinkwrapped version of the multilingual Corel WordPerfect for Linux Personal Edition has a retail price of $69.99 US. A multilingual Server Edition of Corel WordPerfect 8 for Linux is also available from the VAR channel and corporate

resellers for a retail price of $495 US for the full version and $249 US for the upgrade.

Contact: Corel Corporation, 1600 Carling Avenue, Ottawa, ON K1Z 8R7, Canada, Phone: 800-772-6735 (outside North America, +353-1-706-3912), Fax: 613-761-9176, URL: http://www.corel.com/.

NUOPT for S-PLUS

MathSoft, Inc. introduced NUOPT for S-PLUS, an optimization software package for data analysts, researchers and quantitative financial analysts. Key features and benefits of NUOPT for S-PLUS include efficiency and robustness, integration with S-PLUS and a wide range of applications. List price is $9995 US for UNIX and Linux. NUOPT for S-PLUS was developed by Mathematical Systems, Inc., 10F Four Seasons Bldg., Shinjuku-ku, Tokyo 160-0022, Japan.

Contact: MathSoft, 101 Main Street, Cambridge, MA 02142, Phone: 800-569-0123, Fax: 617-577-8829, URL: http://www.mathsoft.com/.

Archive Index Issue Table of Contents

Advanced search

Product Review: The K Desktop Environment, Version 1.0

Bill Cunningham

Issue #59, March 1999

At the risk of sounding like a surfer, KDE was the sharpest, coolest window display I had ever seen on any operating system.

- Developer: KDE Project Team
- URL: http://www.kde.org/
- Price: free
- Reviewer: Bill Cunningham

Everyone's done a double-take at one time or another. For me, they can be the result of seeing a European sports car, a pretty girl or a UFO. I did a double-take on catching sight of a Linux window manager recently.

About a month ago, I received a S.u.S.E. flyer in the mail. The flyer had a screenshot of a new window manager called KDE, the K Desktop Environment, up and running. At the risk of sounding like a surfer, KDE was the sharpest, coolest window display I had ever seen on any operating system. I wanted my Linux desktop to look exactly like that.

Figure 1 is the screenshot that sold me. Compared to KDE, my old standby, FVWM, looked downright humble. I vowed to get KDE as soon as possible.

Figure 1. KDE Screenshot

A Yahoo search led me to the KDE project's page at http://www.kde.org/. The web site has extensive documentation that I won't recite here. A slide show at http://www.kde.org/kdeslides/ summarizes the whole project quite nicely.

What is it?

KDE is primarily a complete, windows-based graphical operating environment for UNIX platforms. The KDE Core, also called **kdebase**, comes with a highly

configurable window manager, control panel, file manager and virtual terminal. A user can download and install just the core package and be up and running in fine style. All your original X Window System applications will still work as before.

However, to fully benefit from KDE, one can additionally install specialized, interoperative application suites from any or all of these functional areas:

- multimedia
- graphics
- utilities
- games
- networking
- system administration

More will surely appear in the future.

These applications were designed to be highly interoperable. At first, I wanted only the basic desktop. It proved to be so powerful and easy to use that I quickly decided to get the multimedia package. These applications work equally well. Inevitably, I will have all the KDE packages by Christmas.

First Impressions

In my experience so far, KDE seems to be quite solid. The basic desktop takes about five seconds longer to come up than FVWM did (I have a P-133, 32MB RAM). Once up, however, the applications run noticeably quicker than comparable, older applications under FVWM. The applications are sharp-looking and responsive. Memory usage seems to be no more than non-KDE applications use. On my system, I had Netscape, the Applix word processor, one kvt and an active PPP connection running simultaneously before I saw any swapping. Even then, I had about 66% of my 16MB swap space still free. This is about the same system usage required by FVWM.

Aesthetics

Working with FVWM is sort of like ice skating on a vast, frozen lake. With KDE, I can have fun on my computer for hours changing the background pattern, moving scrollbars around and fiddling with the controls. Sure, all this was possible with FVWM, but who ever had time to figure out how? With KDE, every aspect of the desktop's appearance is configurable with a couple of mouse clicks.

Who is Shipping with KDE?

Linux distributions from all over the world are now shipping with KDE. Here are some of the latest:

- German companies S.u.S.E GmbH, Delix Computer GmbH and Chip Extra Magazine
- Caldera OpenLinux 1.2
- PTS Linux
- Eurielec 98 and COX-Red Hat 5.0 (Spanish)
- Dream (French computer magazine)
- Turkuaz (Turkish Linux distribution)
- Connectiva (Brazilian-Portuguese)
- Linux Mandrake
- Walnut Creek's FreeBSD 2.2.7
- InfoMagic
- Stampede
- LinuxPPC 1998 (PowerPC)
- MkLinux (Power Macintosh)

New Linux users will probably use KDE as their X interface from the beginning. The following information is for users who would like to switch to KDE from another window manager.

The only warning I would make at this point is that KDE is somewhat large. Together, kdebase and kdelibs take up about 20MB once installed. The kdesupport package was another 4MB installed, and multimedia added 5.4MB. For most hard drives, these numbers are a drop in the bucket. I would even go so far as to say that KDE is worth buying a new hard drive for, if yours is getting full. With a complete Slackware installation, a whole year's worth of FTP downloaded applications and KDE, my 1.5GB hard drive is only 25% full.

Getting the Sources and Installation

The KDE packages can be downloaded from the KDE web page. The web page contains directions for installation. When you unpack the source files, be sure to read the README and INSTALL files as well.

Several mirror sites around the world provide optimum download times. The packages are available in source and binary RPM, source and binary .tgz and source and binary Debian.

The KDE team recommends RPM for the inexperienced UNIX user whose system supports that format. My Slackware system did not, so I had to compile the sources. Although this took about an hour, the process was well-documented and I had no problems with the installation. If you're compiling the sources yourself, here is a tip that may help: decide on and create a “kde root” directory, for example /usr/local/kde. Put your distribution files in /usr/local and use **tar** to unpack them in that directory. All your compiled files and libraries will end up under /usr/local/kde/, and any additional packages you install later will be able to find the necessary libraries and binaries.

Starting KDE the First Time

Once KDE is properly installed, you must create a mechanism to start it. On most Linux systems, **startx** starts the X Window System and then runs another script, /usr/lib/X11/xinit/xinitrc. This script is a link to one of several scripts that start the different window managers such as FVWM, FVWM95, TWM, etc.

Edit the file that /usr/lib/X11/xinit/xinitrc points to and find the line that launches your old window manager. Comment out that line and add a line under it to launch **startkde**. The last few lines of your file should look something like this:

```
# extract from /usr/lib/X11/xinit/xinitrc.FVWM
if [ -f $usermodmap ]; then
 xmodmap $usermodmap
fi
# start some nice programs
xsetroot -solid SteelBlue
#FVWM <-comment this out
startkde # <-and add this!
```

Now, save this file and at the prompt, type **startx**. KDE should fire right up, greeting you with a very impressive deep-blue desktop. Try the virtual desktop selector on the bottom bar. It has four buttons, named one, two, three and four. These switch between virtual desktops, each of which has a different background. The background files are in .jpg format and can be easily changed. I have some .jpg files containing photos taken by the Hubble telescope, which make great backgrounds.

Now Run Something

Now that KDE is up and running, let's try doing something with it. Put your cursor on the bottom bar icons, but don't click anything. After a second or so, a label will pop up with the icon's function. All the way to the right is the “Terminal Emulation (kvt)” icon. **kvt**, or K virtual terminal, is KDE's version of the xterm. Clicking once on this icon will open up a kvt. Don't double-click—that would open two kvts.

An interesting property of the kvt is that it is not by default a login shell. In other words, none of the commands in any of your login scripts will run when you open a kvt: no DIRCOLORs, no aliases, no special environment variables. This is quite easy to change if you desire. Simply click the right mouse button on the kvt icon and open “Properties”. On the “Permissions” tab, make sure you have the **Read** and **Write** buttons pushed in for User. Then on the “Execute” tab, Execute input area, the default command to run is

```
kvt -caption "%c" %i %m
```

To open kvt as a login shell, just add **-ls** so it now reads:

```
kvt -ls -caption "%c" %i %m
```

Then click on “OK”. Your next kvt will open as a login shell. If this doesn't work, shut down KDE and restart it as root. This time, the modification will definitely work.

After having set kvt up as a login shell, you may notice a curious message on the first line of the kvt display that reads:

```
/dev/ttyp2: Operation not permitted
```

Below this line will be your normal shell prompt. This message can safely be ignored.

Create an Icon on the Desktop (Netscape)

In all my years of running X, I never figured out how to open FVWM with icons on the screen. A Netscape icon can be created with KDE in about a minute and doesn't require reading man pages.

On the left border, open the “Templates” folder. Select **File**->New**->**Program. In the KFV dialog's “General” tab, change Program.kdelnk to Netscape.kdelnk. In the “Execute” tab, type in the path to the Netscape executable and specify an appropriate working directory. Click on the icon that seems logical. Click **OK**, and you are done. (See Figure 2.) Remember, don't double-click on the Netscape icon unless you want two browsers to open.

Figure 2. Netscape Icon in KDE

State Preservation at Shutdown

With FVWM, several programs could be opened and iconified in order to set up the desktop. Then, when FVWM exited, it all disappeared and the next time

FVWM was opened, all changes would have to be redone. KDE opens in exactly the same state that you leave it. To me, this is both logical and convenient.

Conclusion

What about KDE's drawbacks? So far, I haven't found anything with KDE that I don't like or that is “broken”. It seems to be solidly engineered and stable. I'm keeping it!

One objection to KDE is that it looks a lot like MS Windows 95. Once you use it for a while, though, you realize KDE is not much like Windows 95 at all. It does have a silver bar along the display bottom (and top), and icons on the left side. However, every aspect of KDE's appearance is configurable; these are just what come out of the box. Similarities to Windows 95 end at the screen pixels.

Some have also said that KDE represents a moving away from the low-level workings of the operating system. For many people, this is actually good news. For the programmers and kernel hackers, Linux is still underneath it all. I believe most will see KDE as a breath of much needed fresh air. Recall what happened to OS/2—a highly specialized operating system that catered strictly to intellectuals.

The Linux community can't simply find a comfortable niche and stay there forever. We are either attracting users or losing them—not everyone is a kernel hacker or system programmer. If Linux is to be a vibrant, mainstream, “world dominant” operating system, it needs conveniences for the average user: straightforward installation, good applications, good looks and ease of use. KDE is a quantum step in this direction.

Gunnery Sergeant Bill W. Cunningham, U.S. Marine Corps, is a system administrator with the Second Marine Aircraft Wing, G-7, Cherry Point, NC (the greatest job in the universe). He likes playing guitar, reading, driving and spending as much time as possible with his wife and four kids. He can be reached at bill@mfs.usmc.mil.

Archive Index Issue Table of Contents

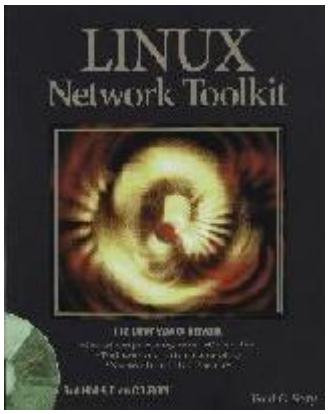
Advanced search

Book Review: Linux Network Toolkit

Russell J. Dyer

Issue #59, March 1999

This very thick and well-written book covers just about every aspect of Linux networking, not just integrating with Windows 95.



- Author: Paul G. Sery
- URL: http://www.idgbooks.com/
- Price: $49.99 US, $69.99 CAN
- ISBN: 0-7645-3146-8
- Reviewer: Russell J. T. Dyer

Linux Network Toolkit by Paul Sery is a tutorial-style computer book that teaches the dedicated reader how to set up and maintain an integrated network composed of a Linux server and MS Windows 95 clients. At the core is, of course, Samba. This very thick and well-written book covers just about every aspect of Linux networking, not just integrating with Windows 95. Various qualities of the book make it superior to other Linux books of this type, but it still adheres to a few Linux book conventions that are bothersome. The price precludes it from being a casual purchase, but for the right Linux users, it is a worthwhile read and investment.

The most impressive aspect of this book is its structure. Mr. Sery delivers a feeling of accomplishment in the first chapter and builds with each successive chapter. Chapter 1 covers installing Linux on a computer, connecting it to a Windows 95 network and configuring the Samba basics. This is no small challenge and his instructions are fairly straightforward if the reader is computer savvy and willing to follow directions. If it doesn't work, the second chapter on troubleshooting should help. With success in hand, Sery goes back and explains Linux and its history and gives recommendations on learning more (i.e., through various web sites, books, etc.). In the following part, Sery returns to the Samba and Linux configurations in much greater detail. He also includes a chapter on putting a Linux server on the Internet, while dedicating another chapter to sharing his skills as a network administrator. He ends the book by returning to the beginning and guiding the reader through the steps of setting up a complex Windows 95 network with a Linux computer as its server. With this organizational structure, Sery provides a very effective tutorial on Linux networking.

At a price of about $50 US, however, *Linux Network Toolkit* is a little costly for the average Linux user. On the other hand, the average Linux user doesn't have a network at home and probably doesn't get to use Linux at work. Those who do use Linux at work might find it helps them advance in their job and pay, perhaps convincing the boss to switch the office server over to Linux. In my case, I was allowed to convert my computer at work to Linux as long as I could interact with the network and not be a burden to others. I met those requirements primarily with Samba and WordPerfect for Linux. So, a book that helps the reader integrate Linux into his office network is worth the cost.

Linux Network Toolkit is nearly 600 pages long and comes with Red Hat 5.0 on CD. Some of the bulk seems unnecessary: about 80 pages of appendices, 20 pages of Linux and related history and a scattering of directory listings, standard scripts and screen shots. The appendices cover **vi** commands, the GNU General Public License, copyrights, a list of web sites and a description of the contents of the CD. I could have done without these and wish they had been written on the CD.

I find Linux's history to be very interesting, but I don't understand why almost every Linux book insists on including it. Sery's accounts of the origins of Linux, UNIX and Samba are well-written, so although I fault him for including histories, I prefer his histories to those in other Linux books I own.

As for directory listings and scripts, some take up to two pages. Granted, they can be necessary, but many times it is sufficient to tell the reader to type the commands and see what happens.

Basically, the publisher, IDG Books Worldwide could have eliminated about 150 pages and the CD and lowered the price. After all, how many copies of Linux do I need? A lower price would have been more palatable and allowed for more sales to make up for the per-book profit.

Sery's approach suggests that he assumes the reader has a fairly good knowledge of computers, but not necessarily much on Linux, whereas the back cover suggests a reader level of intermediate to advanced. This is probably more reasonable. I find the book to be more appropriate for the Linux user who has long since graduated from the “new user” category and is ready to leave the “beginner” mode. It is also useful for the more advanced user who has a few gaps in his knowledge or has never attempted a Linux network. For these users, the book may work just fine as a tutorial or a reference. Along those lines, at the end of each chapter, Sery provides well-thought-out chapter summaries. Most books provide these, of course, but they are often poorly done. Sery's summaries are very effective in reinforcing what the reader just read and later serving as a reference source.

The style of writing is in a somewhat standard form for computer books, but more accessible. Sery occasionally writes in a conversational tone. For instance, at times he uses words like “stuff” and “great”. He doesn't hide from the reader, either. He uses the pronouns “I” and “me” quite often. He also steps out of character occasionally and lets his natural enthusiasm for computers and Linux show through. He tells the reader how he found certain aspects of Linux difficult and confusing when he was first learning it; he then attempts to resolve these problems for the reader. As part of this, Sery sometimes explains in simple terms, step-by-step, the processes going on behind Linux commands. This, to me, is an excellent way to write a useful and approachable Linux book. It also makes it more enjoyable to read. After all, it is about Linux, not stuffy old UNIX. It belongs to us, not rigid institutional companies looking to make a quick billion.

Another accessible feature of Sery's book is the fact that he compares commands in Linux to their counterparts in DOS and Windows. Most Linux books don't do much of this, I believe, because the writers are either coming from a UNIX background or they are anti-Microsoft. If a reader comes from a UNIX background, he probably won't need most of the books written about Linux. Ignoring all that most readers have learned from the Microsoft environment is counter-productive. When explaining the **ls** command, does it really hurt to say that it's like **dir** in MS-DOS? Where appropriate, Sery provides these useful links to what the reader likely already knows. In so doing, he helps the reader learn faster and appreciate the greater flexibility and power of Linux.

One other warning: this is definitely a pro-Red Hat book. I mention this only because before I was a Red Hat user, I hated to spend $30 or more on a Linux book, then find out one-fourth of it was a Red Hat manual. *Linux Network Toolkit* comes with Red Hat 5.0 and contains many comments on how Red Hat is configured. If I were a Debian user, I would probably say it contains an excessive amount of comments on Red Hat. One plus for users of other distributions that use **rpm** is that it provides information on several rpm commands the Red Hat manual.

In conclusion, I would recommend *Linux Network Toolkit* to intermediate Linux users who are constructing and maintaining an integrated Linux network. The price is a little high, but it can be justified for certain readers and needs. The book is partial to Red Hat and has the usual Linux book fillers. All in all, though, it is a well-written, approachable tutorial on Linux networking.

Russell Dyer works for a UNIX-oriented software company, Information Management Consultants (www.imcsite.com), in New Orleans as a systems implementation coordinator. He earned a bachelor's degree in Fine Arts & Humanities from Loyola University in New Orleans. Currently, he's working on a master's degree in English from the University of New Orleans. He's married to his lovely accountant, Shirley Toups Dyer, and they have three children: Kenneth, Geoffrey and Marie. His personal e-mail address is russell@dyerhouse.com.

Archive Index Issue Table of Contents

Advanced search

Linux and the Euro Currency: Toward a Global Solution

Guylhem Aznar

Issue #59, March 1999

Mr. Aznar talks about problems and solutions to adding the euro symbol to the keyboard.

Starting in 1999, each European be using this new currency in his everyday life. Even if cash is not yet available, most prices, wages, invoices, etc. will be labelled in euro as well as in national currencies.

Figure 1. The Euro symbol

In 2002, each local currency except for the British pound (£), Danish krøner (kr), Swedish Kroner and Greek drachma will be replaced by the Euro. This is a major move for a real European union, which will create a market of 300 million people, from Finland to Portugal and Ireland to Greece.

Figure 2. Various currency symbols

Euro and year 2000 support are currently the biggest problems for computer scientists. On UNIX systems where a kind of Y2K bug will not occur until 2038, the euro is the major problem. On Windows machines, Y2K is the major problem since Microsoft has already integrated support for the Euro in Windows 98 and NT. A new proprietary norm was created, called cp1252, which makes these systems the first to properly support the Euro.

Figure 3. Key table with the Euro symbol

The transition from local currencies to the euro also implies mathematical conversions in financial programs: one euro equals 6.55957 French francs, for example. One euro is approximately one dollar U.S. The European Community created a “Euro Workshop” and “Euro Project Team” to find any possible solution for all types of platforms, even cellular phones. Most of the documents

can be found at http://www.stri.is/TC304/Euro/. I used some of their euro symbol pictures to illustrate this article.

Problem

UNICODE was a good candidate for supporting national characters, but X Window System font management would have made it a pain to implement. For example, see plan9 difficulties at http://www.gh.cs.usyd.edu.au/~matty/9term/. Moreover, the numeral console-mode applications could not take advantage of UNICODE: even today, video-card memory is too tight to hold this wide range of characters. Clearly, a solution was needed for UNIX and other 8-bit systems.

A possible solution was creating yet another encoding format for 8-bit systems, which even today do not support the whole range of national characters. For example, it is impossible to create an “oe ligature” or an “s caron” in iso-8859-1, even though they are widely used in several European languages:

- “oe ligature”: the French words boeuf, coeur, oeil, the German poet Goethe
- “s caron”:
  Š
  koda cars

I can hardly imagine a computer where not all 26 alphabetic letters were available, or where u had to be used in place of v because it had not been implemented in its character set. Yet many non-English speaking people face this problem each day when using keyboards without characters such as the “oe ligature” (

œ
) or “s caron” (
Š
).

A New Encoding

Taking advantage of this situation and lack of the euro symbol in iso-8859-1 latin1 (ISO 8859-1 latin1 is the UNIX encoding format, i.e., a correspondence between letters and codes, also used for email encoding), a new encoding format project was started, which would support:

- the euro
- “oe ligature” in both upper case and lower case
- “s and z caron” in both upper case and lower case

- “y diaeresis” in upper case only since (ÿ) already exists in latin1

To avoid breaking compatibility with the widely used iso-8859-1 latin1, it was decided that this new encoding would be based on it. However, there was still a big problem—each of the 255 possible characters existing in an 8-bit encoding had already been assigned in iso-8859-1. Therefore, it was decided that less frequently used characters should be removed to allow integration of the missing ones. In addition, the following changes were made:

- The euro replaced the “international currency sign” (¤): the symbol the former Soviet Union had wanted to replace the dollar sign in 7-bit ASCII encoding format.
- “oe” and “oe ligature” replaced the one-half fraction mark (½) and one-quarter fraction mark (¼), respectively.
- “s caron” and “S caron” replaced “broken bar” (¦) and “floating diaeresis” (¨), respectively.
- “z caron” and “Z caron” replaced “floating acute” (´) and “floating cedilla” (¸), respectively.
- “Y diaeresis” replaced “three quarter” (¾).

None of the replaced characters is printed or used in any national keyboard except for the international currency sign (¤) in French and Belgian ones: this would be a minor loss. Moreover, floating accents only serve the need of “compose”: for example, a circumflex accent (^) can be put onto an e to give è. On French-Canadian keyboards, the cedilla (¸) can be put onto a c or a C to get a ç or a Ç.

Since the cedilla (¸) looks like a comma (,), the diaeresis looks like a double quote (“) and the acute looks like a single quote ('), they could easily be replaced; compose tables would need a minor update. For example, on French and Dutch keyboards, the (”) key is used to get äëïöüÄËÏÖÜ but not “Y diaeresis”, which is missing. One must simply press it, release it (nothing happens) then press a vowel key to get a “vowel diaeresis”.

If there is any other character but a vowel, (“) shows up. Here, adapting to the new encoding would simply mean replacing the latter case by a (”) and adding composing possibilities for (sSzZ). This new encoding is called iso-8859-15 latin9 (nicknamed latin9) to show it is 97% compatible with iso-8859-1 latin1. By the time this article is published, it should already have been approved by ISO.

Implementing iso-8859-15 latin9

Linux Console

Implementation for the Linux console was quite simple. I asked Ricardas Cepas (rch@pub.osf.lt) which tool he implemented his fonts with. He provided me with a custom version of the **chedit** font editor for linux-console. I simply took latin1 fonts and replaced the old unused characters by latin9 new characters, for each latin1 font size (from 16x16 to 08x08), but then I was able to display only iso-8859-15 latin9 characters.

Keyboard

Among the recommendations of the European Commission is "**AltGr-e** should be used to get the euro symbol". (The **Alt** key to the right of the keyboard must be remapped to **AltGr**.) **AltGr** is used as a modifier like **Shift** on German keyboards, @ can be obtained with **AltGr-q**, on French keyboards # is **AltGr-3**, etc. **AltGr** is used in the Linux version of many European keyboards to output 8-bit characters, as a remembrance of things past: there was a time when making dead keys work was impossible. On French and Dutch keyboards, (") and (^) are such dead keys: they act like compose plus this key on the following character. Since many words use (^) or (") (ètre, aigü...), **AltGr-v**, where v is the appropriate vowel, and **Shift-AltGr** vowel were used to get, respectively, vowel-circumflex and vowel-diaeresis. Nowadays, dead keys work with most of the programs except for Netscape or Applixware so these shortcuts are still very much appreciated.

If **AltGr-e** already outputs è, where could the euro be placed?

This hard problem is yet to be solved by any keyboard maintainer; for the French one I am in charge of, I decided to rearrange the "dollar" ($)/ "British pound" (£)/ "international currency symbol" (¤) keys. Since none of these symbols is an official French money, I changed it to "euro"/"eurocent"/"dollar"/"British pound" respectively normal, **Shift**, **AltGr** and **AltGr-Shift** state. However, French keyboard official standard will use **AltGr>-e** for euro, so I had to change back this key, remove unavailable international currency symbol and find a new home for "e circumflex", which was in **AltGr-e** under Linux. I decided to put dollar/British pound/eurocent/e circumflex on this key, only ¤ and ê had to be moved. This was the best possible solution I could imagine, but I am still looking for another solution to ease euro accessibility and keep 8-bit characters shortcuts.

I also added the other latin9 characters, in **AltGr-Shift** state for the unused ones which can also be obtained by ^-sSzZ or "-Y, and in unshifted plus shifted state for the very common

œ

ligature in French.

Figure 4. French Keys

On the screenshot, you can see a representation of the French keyboard with all shortcuts shown: “MAJ” means “Caps Lock”, “Ferme” means “Close”, “Arrèt défil” means “Scroll Lock” and “Con” keys are the extended PC 105 keys (also called “Windows key”) mapped to previous console, next console and last console. This ASCII art is included in fr-latin9.map key map to remind users where all the Linux-specific shortcuts are.

Now, with a font and a key table, it was getting more interesting. I started a beta-release program to get feedback, which was mostly positive. The only drawback was ê; some people wanted it to stay where it was so I showed them how to edit the key table to do this.

X Window System

X fonts are not covered by GPL, while the rest of the package was going to be released under GPL. Since I could not find any iso-8859-1 latin1 X font with a GPL, I had to use an XFree copyrighted basis. With Mark Leisher's **xmbdfed** (mleisher@crl.nmsu.edu), I could very easily change some fonts. X window fonts are *not* scalable; you have to choose Adobe or True-Type fonts (with **xfdtt** for the latter). Also, I didn't feel like editing all the fonts included with X. No real solution exists at present, except switching XFree to True-Type fonts, which would be a good idea since True-Type fonts are scalable, of good quality, many are released under GPL, BSD or public domain licenses and they already support the whole range of latin1 *plus* latin9 thanks to the cp1252 proprietary format.

Resources

Guylhem Aznar is studying medicine at Purpan University Hospital in Toulouse, France. He enjoys swimming, playing chess and listening to music when not writing Linux HOWTOs. He can be reached via e-mail at guylhem@barberouge.linux.lmm.com.

Archive Index Issue Table of Contents

Advanced search