E-mail as a Tool for Sharing Binary Files among Scientists

Vince Hamner, Ching-Wan Yip, and Raymond E. Dessy*

Chemistry Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0212

Received July 2, 1993●

The use of plain-text encoding and decoding software allows binary data files to be successfully transferred between sites via an E-mail pathway. The following paper examines the mechanics of the public domain UUENCODE and UUDECODE software combination. A short tutorial allows the reader to apply the algorithms to sample data files. An understanding of the nature of the utilities and their limitations may allow one to avoid potential pitfalls associated with their use. Many of the commonly used encode/decode software packages are cited, and potential sources of these public domain utilities are provided.

INTRODUCTION

Although an occasional reader will be familiar with the encoding and decoding process required for the safe transmission of binary files through E-mail channels, the actual mechanics of the encode/decode algorithms might not have received careful consideration. Additionally, there will exist a number of working chemists who are presently unaware of this technique. Only recently has our own research group discovered these valuable software tools which ensure the successful transmission of binary files from site to site. In addition to sharing such files internally within our department, it is now possible to safely share large binary (or *image*) files with other sites via E-mail. A few examples of such files might include molecular modeling graphics, scanning electron microscopy (SEM) images, and magnetic resonance imaging (MRI) scans. In the laboratory setting, it is common for many analytical instruments to save their raw data in a binary format. Executable programs (.exe), spreadsheets, and specially formatted documents prepared with a word processor should also be treated as binary files.

Most academic sites and many corporate research centers have realized the importance of E-mail and have readily adopted it as an everyday communications medium. Although it is outside the scope of this paper to discuss the use of E-mail, it is safe to say that it is an integral part of our information technology toolbox. E-mail was designed for the purpose of allowing users to easily share text (or ASCII) files among one another; however, the exchange of binary data files between sites using an E-mail facility is not so straightforward.[1] Luckily there are a few basic steps that can be taken in order to ensure that the integrity of those files will be maintained. The following tutorial was written with that goal in mind.

E-MAIL AND THE STANDARD CHARACTER SET

Short E-mail notes are typically created on-the-fly using mainframe or PC-resident text editors. These editors allow the user to access all of the basic characters that are needed for written English communication (ASCII characters 32 to 126). Perhaps a researcher wishes to prepare a scientific paper using a word processor. How can that file be safely shared with colleagues through an E-mail communications channel? If the user is able to save the document in a text-only format, then the file should transfer via E-mail with no difficulty. Of course, this would be a rare instance as most scientific documents will contain special characters and graphics that must be retained and shared as well. If that is the case, then care must be taken to ensure that the message is *encoded* into a format suitable for transmission as purely a text-based file. Why is this necessary?

One of the powerful features of the word processor rests with its ability to carefully maintain any special formatting information desired by the user. This formatting information will often be found in the header and footer portions of the document. When viewed with a simple text editor, that information is generally seen by the user as a series of meaningless, yet special, characters. As mentioned earlier, most word processors will also allow the use of an extended character set. These characters might include mathematical symbols, bullets, arrows, hearts, smiley faces, and so on. Such special symbols, sometimes referred to as *nonprintable* characters, typically pose little threat to the recipient of the E-mail message (ASCII characters 0 to 31 and 127 to 255). Often they will be mapped back to a character within the standard character set and, if so, will only arrive as gibberish at the destination site. The E-mailed text should still be readable. However, the word processor at the receiving end will be unable to successfully interpret the necessary formatting information. Any corrupt mathematical symbols, graphics, or other special characters in the body of text would require manual replacement.

What about the SEM data or the executable (.exe) file that we wish to share? In order for the integrity of those files to be maintained, each individual character must successfully arrive intact at the receiver's site in the exact same form that it was transmitted by the sender. Otherwise, the images delivered will be nonviewable; any programs sent will not run; and important data files will have been corrupted. As these files are not manually repairable, the sender will soon be advised of their negligence by the intended receiver. Certainly, special characters will be unnecessary, but be forewarned that *special words* may be invoked!

MATHEMATICAL SOLUTION TO THE PROBLEM

Assume that we have a binary data file that we wish to E-mail. By definition, a byte is an 8-bit binary number. Therefore, there exist 2^8 or 256 possible decimal numbers which can be represented by a sequence of 8 binary numbers. These 256 possibilities conveniently allow for the participation of 256 possible characters in the ASCII set. In order for the file to be transferred through any E-mail system in a completely unaltered state, it can only consist of the printable characters (ASCII 32 to 126). Thinking purely in mathematical terms, a group of three bytes in our data file could be represented

* Author to whom correspondence should be addressed.
● Abstract published in *Advance ACS Abstracts*, March 15, 1994.

Chart 1

a

c	h	e	m	i	s	t	CR	LF
01100011	01101000	01100101	01101101	01101001	01110011	01110100	00001101	00001010

b

0011000	0110110	0100001	0100101	0011011	0010110	0100101	0110011	0011101	0000000	0110100	0001010
0100000	0100000	0100000	0100000	0100000	0100000	0100000	0100000	0010000	0100000	0100000	0100000
0111000	1010110	1000001	1000101	0111011	0110110	1000101	1010011	0101101	0100000	1010100	0101010
8	V	A	E	;	6	E	S	=	`	T	*

as a 24-bit binary number. That same 24-bit number could also be represented as a series of four 6-bit *packets* [$2^6 = 64$ decimal numbers]. By regrouping the bits, the complete ASCII character set 0 to 255 is mapped by using only 64-bit patterns. However, we cannot use numbers corresponding to 0 to 63 decimal because the first 32 combinations represent nonprintable characters. They might not be transmitted through conventional E-mail systems intact.

However, what would happen if an offset number were added to the new series of 6-bit packets? An offset such as 32 (decimal) would map the 6-bit packets to the numbers 32 to 95 which correspond to printable ASCII characters. This range is now agreeable (greater than 31 and less than 127). One problem remains. The number 32 represents the ASCII character for a blank space. Since E-mail systems often ignore trailing blanks, all numbers mapped to 32 are replaced by 96. The full mathematical operation, including original offset, is represented by ((0 + 32) + 64). The ASCII number 96 corresponds to the single open quote character (‘). This operation will ensure that there will be no *blank spaces* within the encoded text file that might be ignored during the E-mail transfer.[2] The resulting set of *printable* characters follows:

‘!"#$%&'()*+,-./0123456789:;<=>

?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_

In the mapped, encoded file, each 6-bit packet can be represented by one of the above printable characters. We have successfully created a cryptogram that will allow a safe transmission of the file to our colleagues via E-mail. The characters will be received in exactly the same form as they are sent. The scientist or editor on the receiving end will acquire an intact and uncorrupted data file at their site. It is now necessary to share the algorithm with the receiving site so that they may successfully decode the message. Others have already solved this problem by including the algorithm in shareware utility programs that perform these encoding and decoding tasks. These two pieces of software, commonly referred to as UUENCODE and UUDECODE, are presently available in the public domain and are readily accessible by all.

Example 1. We consider a text file called *test.txt* which has the word *chemist* in it. It consists of six ASCII characters and a return (which is actually a combination of carriage return and line feed in DOS).

After the file *test.txt* passes through UUENCODE, the resulting file *test.uue* will look like this:

```
begin 644 test.txt
) 8VAE; 6ES=‘T*
‘
end
```

The first line starts with the word *begin*. It is followed by a file mode setting *644* which is useful on those systems that implement file security (UNIX, for example). The name of the original file is included at the end of this line.

The body of the file is the UUENCODEd message itself. The first character serves as the character count. A ")" is ASCII 41, which means there are nine characters in this line (41 minus the bias 32). Typically each line in a UUENCODEd file will contain 60 characters (60 × 6 = 360 bits). As a result of the translation from 8-bit *bytes* to 6-bit *packets*, this corresponds to a 45 character segment of the original file (45 × 8 = 360 bits). Generally the letter "M" is observed as the leading character count. M corresponds to ASCII number 77 or number 45 in UUENCODEd form (77 – 32 = 45). The final line in the UUENCODEd file is usually the exception to this rule. It will typically contain less than 60 characters.

The ASCII characters in the test file have the 8-bit representations given in Chart 1a.

By combining every three 8-bit bytes, rearranging them into four 6-bit packets, and adding the offset 32 (binary 100000), the encryption in Chart 1b is achieved.

A line with a single open quote (‘) indicates that a line with zero code follows. The .uue file always closes with the word *end*.

Try to reverse the encryption process in order to decode the following sample file.

```
begin 644 test2.txt
M0V]N9W)A='5L871I;VYS(2‘@66]U(&AA=
        F4@;6%S=&5R960@=&AE('5U96YC
/;V1I;F<@<V-H96UE+@T*
‘
end
```

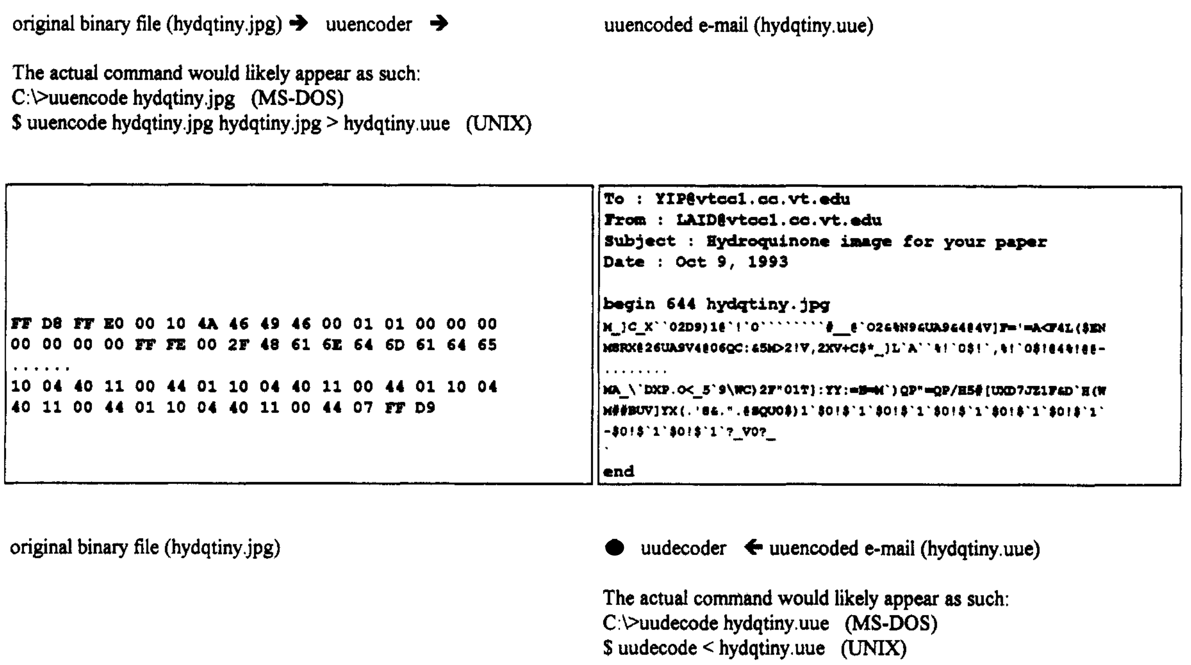
The decoded answer may be found in ref 3.

Example 2. Parts a and b of Figure 1 illustrate the actual encoding process that would be implemented for a molecular modeling image arbitrarily called hydqtiny.jpg. Note the appearance of the contents of the original binary file (*hydqtiny.jpg*) versus the contents of the subsequent UUENCODEd file (*hydqtiny.uue*). The header information found in Figure 1b indicates that the .uue file has already been received as E-mail.

If read in the reverse direction, parts a and b of figure 1 also illustrate the decoding process that would be implemented by the receiver. Following extraction from the E-mail facility, UUDECODE will be applied to the file *hydqtiny.uue* in order to obtain the original image file *hydqtiny.jpg*.

ENCODERS AND DECODERS

The early history of the encoding/decoding software is not well documented.[4,5] Many UNIX users will find that their system's printed manuals (or on-line help files) briefly describe and address the usage of the UUENCODE and UUDECODE commands. Published user's guides written for the UNIX system will occasionally mention these utilities.[6] On the other



original binary file (hydqtiny.jpg) ➔ uuencoder ➔ uuencoded e-mail (hydqtiny.uue)

The actual command would likely appear as such:

```
C:\>uuencode hydqtiny.jpg   (MS-DOS)
$ uuencode hydqtiny.jpg hydqtiny.jpg > hydqtiny.uue   (UNIX)
```

```
FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 00
00 00 00 00 FF FE 00 2F 48 61 6E 64 6D 61 64 65
......
10 04 40 11 00 44 01 10 04 40 11 00 44 01 10 04
40 11 00 44 01 10 04 40 11 00 44 07 FF D9
```

```
To : YIP@vtccl.cc.vt.edu
From : LAID@vtccl.cc.vt.edu
Subject : Hydroquinone image for your paper
Date : Oct 9, 1993

begin 644 hydqtiny.jpg
M_]C_X``02D9)1@`!`0````````#__@`O2&%N9&UA9&4@4V]F=']A<F4L($EN
MBRX@26UA9V4@06QC:&5M>2!V,2XV+C$*_]L`A``%!`0$!`,%!`0$!@4%!@@-
........
MA_\`DXP.O<_5`9\WC)2F"01T]:YY:=B=M`)QP"=QP/H5$[UXD7JZ1F&D`H(W
M##BUV]YX(.'@&."*@$QUO$)1`$0!$`1`$0!$`1`$0!$`1`$0!$`1`$0!$`1`
-$0!$`1`$0!$`1`?_VO?_
`
end
```

● uudecoder 🡐 uuencoded e-mail (hydqtiny.uue)

original binary file (hydqtiny.jpg)

The actual command would likely appear as such:

```
C:\>uudecode hydqtiny.uue   (MS-DOS)
$ uudecode < hydqtiny.uue   (UNIX)
```

Figure 1. (a, left panel) Original binary file. (b, right panel) Uuencoded file embedded in an E-mail message.

hand, those chemists using the DOS operating system outnumber their UNIX-based colleagues by a magnitude of at least four-to-one.[7] The UUENCODE and UUDECODE utilities are typically not distributed with DOS-based systems. Additionally, those users will rarely find preprinted literature regarding the mechanics and usage of the encoding/decoding tools that are pertinent to their own platforms.

A wide variety of compatible software is available in the public domain for each of the prevalent operating systems. The implementations of these utilities are found in numerous formats such as binary executables, BASIC, C, and Pascal versions. Table 1 provides a listing of commonly used encode/decode software packages and suggests potential archive sites that allow downloading via anonymous ftp (file transfer protocol) log-ins.

There are other encoding schemes that use similar mechanisms to convert binaries to *regular* text. One of these is called XXENCODE. The UUENCODE character set utilizes several "special" characters such as

!"#$%'()+,-/:;<=>?@[\]^_

that may be translated improperly during mail transfer on particular systems (for example, during the EBCDIC to ASCII conversion on some IBM mainframes). The XXENCODE algorithm provides a solution to this problem by employing the following character set:

+-0123456789ABCDEFGHIJKLMNOP

QRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Although the XXENCODE algorithm is advantageous as a result of its fail-safe encoding scheme, it has not yet achieved the wide-spread popularity of UUENCODE.

For Macintosh users, a variation of the UUENCODE scheme is more frequently used.[8] Commonly called BinHex (or BinHex4), this format converts every 3 bytes of data into 4 characters in the same manner as UUENCODE and XXENCODE. However, the 64 characters that are utilized now include the following:

!"#$%&'()*+,-012345689@

ABCDEFGHIJKLMNPQRSTUVXYZ['abcdefhijklmpqr

In addition to the raw data, a BinHex file has a header section that contains information pertaining to the Macintosh environment (for example: file name, file type, creator, and file flags). Unlike UUENCODE, there is no need to include the file name in conjunction with the data since it is embedded in the header. Since Macintosh files can contain two distinct sections—a data fork and a resource fork—BinHex was designed to handle each by including both in the encoded file. There are additional useful features associated with BinHex as well. For instance, run length encoding (RLE) is implemented in order to compress repeating characters, and a 16-bit cyclic redundancy check (CRC) ensures data integrity.[9]

POTENTIAL PROBLEMS

File names may be handled differently from computer to computer. As an example, we assume that a researcher works solely on a UNIX system. This researcher (the *originator*) generates three binary files having the following filenames: *benzoic_acid1.bin*, *benzoic_acid2.bin*, and *benzoic_acid3.bin*. They are subsequently UUENCODEd (for example, *benzoic1.uue*, *benzoic2.uue*, and *benzoic3.uue*) and then E-mailed to a colleague (the *receiver*) who works only on a DOS machine. This causes a potential problem. After decoding, UUDECODE will be forced to save the original files using the DOS file convention. DOS only allows eight file name characters followed by three characters of file extension. Following the decoding process, the first file will be saved as benzoic_.bin. A home-brew decoder might save the second and third files as benzoic_.bin as well, overwriting the first two UUDECODEd files. Thankfully, most public domain UUDECODErs will warn the user of a potential file overwrite *before* the second file is decoded to benzoic_.bin again. The simplest solution to the problem would be to pull each of the .uue files into a text editor and selectively edit the file names so that they are suited to the DOS environment.

Table 1. Public Domain Encode/Decode Software and Potential ftp Sites

software	implementation provided by
MS-DOS	
UUENCODE.C and UUDECODE.C	regents of the University of California, Berkeley
UUENCODE.PAS and UUDECODE.PAS	David Kirschbaum
UUDECODE.BAS	Kieth Petersen
UUENCODE.TP5 and UUDECODE.TP5	David Kirschbaum and Toad Hall
UUENCODE.COM and UUDECODE.COM	Theodore Kaldis, David Kirschbaum, and Toad Hall
XXDECODE.BAS	Kieth Petersen
XXDECODE.COM	David Kirschbaum and Toad Hall
XXINSTAL.BAT	David Camp
available via anonymous ftp from sites such as: oak.oakland.edu in: /pub/msdos/starter	
MS-DOS	
UUDO12.ZIP	Ryan Kim
UUEXE522.ZIP	Richard Marks
QUX01_91.ZIP	Theodore A. Kaldis
SUUD10.ZIP	Trevor Ryder
TOADUU20.ZIP	David Kirschbaum and Toad Hall
UUDOALL.ZIP	Howard Helman
UUXFER20.ZIP	David Read
available via anonymous ftp from sites such as: oak.oakland.edu in: /pub/msdos/decode	
MS-DOS, UNIX, and VAX/VMS	
NCDC151.ZIP	Jurgen A. Doornik
available via anonymous ftp from sites such as: oak.oakland.edu in: /pub/msdos/filutl	
MS-DOS and UNIX	
UNC.C	Mark Maimone
available via anonymous ftp from sites such as: vacation.venari,cs.cmu.edu in: /unc-2.3	
WINDOWS	
EXTRCT34.ZIP	Darren E. Penner
WNCODE15.ZIP	G. H. Silva
UUCODE20.ZIP	Bruce Sabalaskey
available via anonymous ftp from sites such as: ftp.cica.indiana.edu in: /pub/pc/win3/util	
MACINTOSH	
UU-CAT-10.HQX	Alan Goates
UU-LITE-13.HQX	Jeff Strobel
UUTOOL-232.HQX	Bernie Wieser
TIGER-11.HQX	Robert Lenoil
MPW-CONVERT-TOOLS.HQX	Sak Wathanasin
BINHEX-50.HQX	Yves Lempereur
available via anonymous ftp from sites such as: sumex-aim.stanford.edu in: /info-mac/util	
UNIX	
UUCONVERT_SOURCE.ZIP	Jeff Wiegly
available via anonymous ftp from sites such as: ftp.rahul.net in: /pub/bryanw/misc	

Another potential problem may involve file size. Some mail systems cannot handle E-mail messages that are longer than 1000 lines. It is not uncommon for large image files to exceed that limit. The simplest way to determine such a limitation would be to create a sample text file and send it first as a test. Another common solution—one that ensures compatibility beforehand—is to split the .uue files into parts that are shorter than the 1000 line limit and E-mail them in small sections. It will then only be necessary for the receiver to selectivity edit the .uue segments and recombine them with a text editor. Additionally, there exist *smart* decoders that will recognize when a .uue file exists in multiple parts (for example: *unc*, *uudo*, *uuexe*, and *uuxfer*). These decoders offer a considerable savings of time in comparison to the task of patching the .uue segments together by hand. They are highly recommended to those users who find themselves too frequently mechanically combining .uue segments. Such a task is better implemented in software.

The .uue example shown earlier does not implement a checksum for error detection. Line checksum, file checksum, and cyclic redundancy check (CRC) can be utilized to ensure the integrity of a file.

As mentioned earlier, if there are lines within a .uue file that end with trailing spaces, some mail systems may truncate them. The integrity of the binary file will have been corrupted. The implementation of UUENCODE discussed earlier uses the single open quote character, ‘, to replace the blank space in order to circumvent such a problem. Be aware that this is a viable, but not necessarily standard, solution to the problem.

Some mail systems handle extremely long lines of text by doing a *line wrap*. Most of the UUENCODE programs use only 45 characters per line (which translates to 60 following encoding) in order to avoid this potential problem.

AN E-MAIL STANDARD INTERCHANGE FORMAT FOR BINARY DATA, GRAPHICS, SOUND, AND VIDEO

An emerging E-mail standard that will include nontextual information is called the Multipurpose Internet Main Extensions (MIME).[10,11] There are seven types of messages supported by MIME:

text	textual information
message	for encapsulating a mail message
multipart	for nontextual, multiple segments of data
image	still image or picture data (JPEG and GIF files, for example)
audio	audio or voice data
video	video or moving image data, possibly including audio data (MPEG files, for example)
application	special binary data[12]

Two encoding schemes will be implemented by MIME. *Quoted-printable* encoding was designed for representing data that are primarily printable ASCII characters. It is considered to be a reasonable compromise between readability and reliability in file transmission. The other scheme is referred to as *BASE64* encoding. Similar to the UUENCODE scheme, the BASE64 algorithm also ensures that nonprintable characters are mapped to characters that may be safely transferred via E-mail channels.[13]

Although outside the scope of this document, details regarding the MIME standard are found in ref 13. The document is available on-line from a number of anonymous ftp sites (/rfc/rfc1341.txt on ftp.nisc.sri.com, for example). A few vendors have already released products that support the MIME standard and other software products are due for release later this year.[14]

It appears that MIME will begin to gain widespread acceptance; however, this is a process that will not occur overnight. In the meantime, most users will continue to safely share their binary files via E-mail in UUENCODEd form. And, now you can as well.

ACKNOWLEDGMENT

We would like to thank the reviewers for their helpful comments.

REFERENCES AND NOTES

(1) Gunnerson, G. Binary Attachments on Public Mail Services. *Network Comput.* **1993** (Aug), 106–114.
(2) Neuwirth, E. *A Short Guide to Networking and File Transmission*; 1990 (Apr 23). Available via anonymous ftp from oak.oakland.edu in /pub/msdos/starter/transgid.txt.
(3) Following the decoding process, test2.txt reads: Congratulations! You have mastered the uuencoding scheme.
(4) Anderson, B.; Costales, B.; Henderson, H. *UNIX Communications*; SAMS—Division of Macmillan Computer Publishing: Carmel, IN, 1991; p 570.
(5) The literature indicates that the UUENCODE/UUDECODE software was first bundled with BSD UNIX 4.0 and subsequently appeared in UNIX systems based on AT&T System V that included University of California at Berkeley (UCB) enhancements.
(6) Gilly, D. *UNIX in a Nutshell, 2nd ed.*; O'Reilly and Associates: Sebastopol, CA, 1992; p 2-114.
(7) Bargo, J. How Today's Chemists Compute. *Today's Chem. Work* **1993** (Sept), 16.
(8) Lewis, P. N. BinHex 4.0 definition, 1991 (Aug). Available via anonymous ftp from sumex-aim.stanford.edu in /info-mac/dev/info/binhex-40-specs.txt.
(9) Crenshaw, J. W. Implementing CRCs. *Embedded Syst. Program.* **1992** (Jan), 18–45.
(10) Baker, S. Mail Migrates to Multimedia. *UNIX Rev.* **1993**, *11* (No. 6), 23–37.
(11) Molta, D. With Super-TCP and Z-Mail, What's Yours Is MIME. *Network Comput.* **1993** (Aug), 40–46.
(12) Knack, K. MIME Silences Multimedia Critics. *LAN Comput.* **1992**, *3* (No. 5), 3.
(13) Borenstein, N.; Freed, N. RFC 1341-MIME: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. *Network Working Group-Request For Comments*: *1341* **1992**, (June), 14–19.
(14) Korzeniowski, P. E-mail Standard Products Appear. *Open Syst. Today* **1993** (Mar 15), 26.