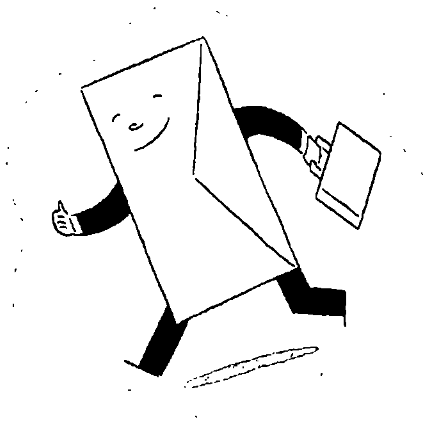
Envelopes

by Jeffreys Copeland and Haemer

In our last column, we promised to show you how to create envelopes for the letters you've been generating using scripts like the ones we illustrated. As we explained last month, our intent in this series is to address real day-to-day business problems and to show you not only how we solve those problems, but also to explain why we solve them the way we do.



The Printer Approach

Most printers will take envelopes, but how they take them varies from printer to printer. This means that what we show you will probably have to be fiddled with a bit to fit your printer. (Don't worry—we'll show you where.)

First things first: What sort of input should the program take? In our last column's solution, we used the `nroff/troff-mm` letter macros for our letters. Following that model, one approach would be to write a separate macro package that does envelopes. This would let us have input files that contained the addresses of the sender and recipient, and we could write shell scripts, similar to our letter scripts, that bring in templates to fill in.

That's a perfectly reasonable approach, but we don't like the fact that it duplicates effort. After all, the letter already has that information: A formal letter normally has both addresses printed inside of it, before the salutation. The `mm` letter macros have `.WA/.WE` and `.IA/.IE`

Jeffrey Copeland (`copeland@alumni.caltech.edu`) *is a member of the technical staff at QMS's languages group, in Boulder, CO. His recent adventures include internationalizing a large sales and manufacturing system and providing software services to the administrators of the 1993 and 1994 Hugo awards. His research interests include internationalization, typesetting, cats and children.*

Jeffrey S. Haemer (`jsh@canary.com`) *is an independent consultant based in Boulder, CO. He works, writes and speaks on the interrelated topics of open systems, standards, software portability and porting and internationalization. Dr. Haemer has been a featured speaker at Usenix, UniForum and Expo Kuwait.*

macro pairs for specifying source and destination addresses. This means that we could just have our shell script extract the addresses from the letter itself. The same file could serve as input to both the program that produces the letter and the script that produces the envelope.

We'll meld these approaches. The script we write will use the letter as its input. But after we extract the addresses, we'll format them for envelopes by writing new definitions for the macros that mark them, and wrap all this up in a shell script. This will give us a chance to discuss how to craft shell scripts.

Scaffolding

We'll begin by constructing a script that doesn't change the letter at all. This lets us get the framework right, and postpones dealing with the details of nroff:

```
#! /bin/sh
PATH=/bin:/usr/bin

ARGV0=$(basename $0)
ROFF=cat
PRINT=cat
RM=:
TMPFILE=/tmp/${ARGV0}.$$
USAGE="doenv|doenv.lh [flags] filename"

abort() {
    echo 1>&2 $*
    exit 1
}

while test ! -r "$1"
do
    case "$1" in
    -?|-:)      abort $USAGE ;;
    *)     ROFF_FLAGS="$ROFF_FLAGS $1"; shift;;
    esac
done

test $# -eq 1 || abort $USAGE

fmt_and_print() {
    $ROFF $ROFF_FLAGS $* | $PRINT
$PRINT_FLAGS
}
# put address & formatting commands into TMPFILE
cp $1 $TMPFILE

fmt_and_print $TMPFILE
$RMv$TMPFILE
```

The #! Cookie

The first line of this script guarantees that it will be interpreted by /bin/sh even if the user is running another shell. This little piece of magic is worth understanding.

When the system tries to execute a program, it begins by opening the file and looking at the first few characters for a "magic cookie," which tells it what kind of file it's trying to execute. If you peek into /etc/magic, you'll see the kinds of things the system might find. (This is actually the file that file uses to take an educated guess at file types.)

If the cookie says the file's in an executable format, the system tries to load and execute it directly. However, if the first two bytes are #!, the system accepts what follows on the line as the name of an interpreter (plus an optional argument) and then uses that interpreter to interpret the file. For example, execute this script:

```
#!/bin/cat
Hello, world
```

Environment Variables

We now set several environment variables. The first of these is the $PATH variable, which we set by habit to discourage Trojan horses. We'll use the remainder of the variables elsewhere in the program.

Notice that instead of hard-coding the names of files and commands, we use variables to hold those names. Not only does this factor out common file names, like $TMPFILE, but it also lets us set $ROFF, $PRINT and $RM to benign commands while we're debugging. In this version of the script (we wrote several), we've set our temporary file, $TMPFILE, to /tmp/${ARGV0}.$$. The variable $$ is the process ID, which makes the output of each run distinct. At this early stage, we set $RM to the null command, :, so that we can examine intermediate results in detail. This leaves a lot of temporary files around, but the way we've named them lets us remove them with the command rm /tmp/doenv*.

It's also worth noting that we've set both $ROFF and $PRINT to cat. This lets us see the output directly on the screen, or lets us discard it, redirecting it to /dev/null.

Other Nuts and Bolts

Following this setup, we do argument processing, using the same general model we showed last month. The while loop looks at each argument until it finds either the name of a readable file, which tells it what to use as input, or a -? or -: argument, which tells it to issue a usage message. (We know of no UNIX command for which -: is a legal argument.)

Next, and as promised, the address extraction and macro rewriting are sidestepped entirely. We just copy the input directly to the temporary file, unchanged, as a place-holder for those steps.

Finally, we format and print the temporary file, using a function we define, and then clean up after ourselves.

Having gotten this far, we can begin to test our code. The commands

```
./doenv
./doenv -?
./doenv /dev/null
```

and

```
./doenv /etc/passwd | diff - /etc/passwd
```

should all yield predictable results.

Nroff-ery

Listing 1 shows the nroff prologue JSH uses to print his envelopes.

There's no need to try to turn you into troff experts here. JSH uses these commands because JLC told him to. JSH coaxed JLC to the Canary Software offices one evening with vague promises of liquid refreshments. We then started with the code JLC uses to print envelopes and tweaked parameters until the envelopes that came out on JSH's printer (a Hewlett-Packard LaserJet 4M) looked reasonable. As long as you have enough liquid refreshment, this isn't a bad approach.

Here are the directives you may need to play with:

```
.in     \" indents text from the left margin
.sp     \" spaces down the page
.ps     \" sets the point size
.vs     \" sets the space between lines
```

If you want to learn how to write nroff macros yourself, Dale Dougherty and Tim O'Reilly's *Text Processing* (Hayden Books, 1987) is excellent.

Putting It into the Script

If this were a lot of nroff source, we might put it into a macro package and install it in /usr/lib/tmac, along with the rest of the macro packages. But because it isn't, we'll put it into the script itself as a "here document." The most typical syntax for a here document is

```
cat > $TMPFILE << EOF
...     # text here
EOF
```

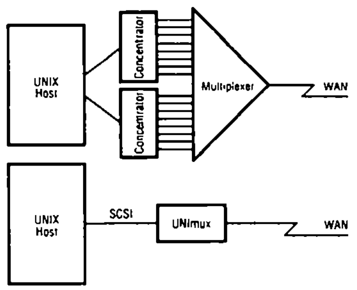
NEW FROM DCB

CLEAN UP CABLE CLUTTER –AND REDUCE YOUR COSTS– WITH THE NEW DCB UNIMUX

Eliminate that snarl of wires to your standard multiplexer. With the DCB UNImux, you can make ***all*** your local connections with a single cable to a SCSI interface and reduce the number of slots used.

Replace this cable mess:

With the UNImux solution:



For AIX, HP-UX, SCO, DEC and SUN.

DataComm for Business, Inc.

Manufacturers of modems, multiplexers, everything for your data communications application.

807 Pioneer, Champaign, IL 61820

Toll-free: (800) 637-1127

n Illinois: (217) 352-320

Circle No. 6 on Inquiry Card

WE BUY, SELL & RENT

RS/6000

PARTS • FEATURES
UPGRADES • CPU'S

- System 36 Conversions
- AutoCad Available for RS/6000
- Sun Sparc Compatibles

New & Used IN STOCK
Complete Technical Center, Installation,
Stock Parts & Features for RISC.

COMPUTER MARKETPLACE

(800) 858-1144

A Publicly Traded Company NASDAQ: MKPL

***MOTOROLA UDS** Western Regional Distributor of the Year*

CDLA IBM®

1490 RAILROAD ST., CORONA CA 91720
TEL (909) 735-2102 • FAX (909) 735-5717

Circle No. 2 on Inquiry Card

Listing 1. Envelope Printing Prologue

```
.de AS \" return address setup
.nr Rp 1
.ps 11
.vs 13
.sp |.2i\" where to start the return address
..
.de WA \" Personal return address
.AS
.in 0.75i
.if \w"\\$1" \\$1
.if \w"\\$1" .ds WN "\\$1"
..
.de LA \" Letterhead return address
.AS
.in 3.0i
.ie \w'\\*(WN' /\\*(WN
.el /Haemer \\*(WN \" for example
..
.de IA \" redefine the IA macro
.ps 14
.vs 16
.sp |2i
.in 4.5i
..
.de DR \" default return address
.if !\\n(Rp \{\
.WA
Jeffrey S. Haemer
Canary Software Inc.
960 Ithaca Drive
Boulder, CO 80303
.\}
..
.em DR
.nf
.ss 18
```

which copies all the text from << `EOF` to `EOF` into `$TMPFILE`.

In our case, there are two wrinkles that we have solved by using slightly more complex variants.

- Much `nroff` syntax would be misinterpreted by the shell, so we need to prevent the shell from interpreting any of the lines in the here document. We can do this by quoting the terminator.
- We want to be able to indent the input text so we don't confuse `nroff` commands with shell commands when we're reading the scripts ourselves. We can do this by using <<- instead of <<, which causes the shell to strip the leading white space.

Put together, we get this:

```
cat > $TMPFILE <<- "EOF"
        ...     # text here
EOF
```

Making It Flexible

We want to be able to print both plain and letterhead envelopes. We'll use the same kind of solution as we did last month: keying on the name of the command.

For plain envelopes, we print a return address in the upper-left-hand corner. JSH's letterhead envelopes have "Canary Software Inc." and his business address printed in the upper-left-hand corner in a pleasing typeface (Goudy Extended). We've also added commands to append `/Haemer` to the first line, the way he might if he were addressing the envelopes by hand so that the mailroom knows who it gets returned to if it comes back (although this is generally not an issue in a company where the president is also the janitor).

We'll use taking these addresses from a file as an exercise for the reader. Putting it all together, we get this:

```
#! /bin/sh
PATH=/bin:/usr/bin

ARGV0=$(basename $0)
ROFF=${ROFF:-groff}
ROFF_FLAGS="-P-l -Tps -mps -fP"
PRINT=lpr
PRINT_FLAGS=-h
RM="rm -f"
TMPFILE=/tmp/${ARGV0}.$$
USAGE="doenv|doenv.lh [flags] filename"
abort() {
        echo 1>&2 $*
        exit 1
}
while test ! -r "$1"
do
    case "$1" in
    -?|-:) abort $USAGE ;;
    *)     ROFF_FLAGS="$ROFF_FLAGS $1"; shift;;
    esac
done

test $# -eq 1 || abort $USAGE

fmt_and_print() {
    $ROFF $ROFF_FLAGS $* | $PRINT $PRINT_FLAGS
}

# put address and format commands into TMPFILE
cat > $TMPFILE <<- "EOF"
    .de AS       \" return address setup
    .nr Rp 1
    .ps 11
    .vs 13
    .sp |.2i     \" where to start return address
    ..
    .de WA       \" Personal return address
```

```
.AS
.in 0.75i
.if "\$1" \$1
.if "\$1" .ds WN "\$1"
..
.de LA        \" Letterhead return address
.AS
.in 3.0i
.ie  /w'\\*(WN ' /\\*(WN
.el /Haemer \\*(WN          \" for example
..
.de IA        \" redefine the IA macro
.ps 14
.vs 16
.sp |2i
.in 4.5i
..
.de DR        \" default return address
.if !\\n(Rp \{\
.WA
Jeffrey S. Haemer
Canary Software Inc
960 Ithaca Drive
Boulder, Colorado 80303
    .\}
        ..
        .em DR
        .nf
        .ss 18
EOF
case $ARGV0 in
        doenv)  sed -n -e '/^\.IA/,/^\.IE/p' \
                             -e '/^\.WA/,/\.WE/p' $1;;
    doenv.lh)  sed -n -e '/^\.IA/,/^\.IE/p'\
                -e '/^\.WA/s/.WA/.ds WN/p' $1;
                echo .LA;;
     *)         abort $USAGE ;;
esac >> $TMPFILE
fmt_and_print $TMPFILE
$RM $TMPFILE
```

So Where Are We?

We'll leave you with a reader exercise: What happens when you have not only a letter but also a bulky enclosure, and you need to send the letter in a 9-by-12-inch envelope? Obviously, you need a stick-on label. Modify our envelope-printing program to print a label on plain paper to be affixed with clear tape to the package.

Next time, we'll deal with another business problem: address books. ▲

CALL FOR
HARDWARE SPECIALS

RS/6000
RT/6150
SERIES/1
SYSTEM/36
AS/400

IBM
Authorized
Distributor Product
Integrator

CALL:
(800) 888-2000

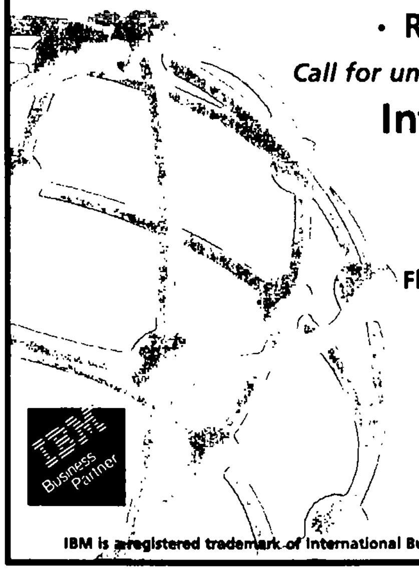
Dempsey
BUSINESS SYSTEMS
Where IBM Quality Is Second Nature.

18377 Beach Blvd., Suite 323
Huntington Beach, California 92648
(714) 847-8486 • FAX (714) 847-3149

IBM is a registered trademark of International Business Machines Corporation.

Circle No. 8 on Inquiry Card

Tomorrow and Beyond....
IBM® RS/6000® and idp
Complete System Integration
- Hardware
- Software
- Connectivity
- RAID



Call for unparalleled service

International Data Products
Minnesota
Florida • California
800.846.7254

IBM Business Partner

IBM is a registered trademark of International Business Machines Corporation

Circle No. 13 on Inquiry Card