

# Work

by Jeffreys Copeland and Haemer



ALEX GROSS

*“Teach us to number  
our days, that we may  
attain a wise heart.”*

–Psalms 90:12,  
Revised Standard

*“A calendar, a calendar!  
Look in the almanac; find  
out moonshine, find out  
moonshine.”*

– William Shakespeare’s  
*A Midsummer Night’s  
Dream*, Act III, Scene 1

## Calendar Pages

There’s a whole aisle of them at your local Staples or OfficeMax. There’s a counter at the boutique stationary store—and in Boulder, even at the kitchen gadget store—devoted to them, complete with a helpful twenty-something employee. We get catalogs from the mail-order fetish fountain pen store that’s mostly taken up with them: calendars. Last month we showed you some interesting `troff` tricks, and because the beginning of the year is a very natural time to talk about calendars, we thought we would continue with a special macro package we’ve built and added to over the years to produce page inserts for those ubiquitous daily planners.

We started down this road to perdition nearly 20 years ago when Copeland’s wife took a time-management course at work. Soon thereafter she started saying things like, “Gosh, it would be really nice if I had a form to go in this notebook for keeping track of projects around the house.” And,

“Y’know, I’d like to have a form for keeping a list of books I’m missing from various mystery series.” Laser printers were just starting to become available but were priced around \$15,000, so one-offs were difficult. However, we had a fabulous high-resolution phototypesetter sitting in the office, and we occasionally managed to turn out a few feet of film from it for purposes of testing the `troff` back end. This meant that we could very easily do multiple copies of a form from a single, very nice, master copy.

(Once again, we missed a business opportunity: At about the same time, as we were knocking off a few dozen copies of useful forms for Copeland’s wife, Day Runner Inc. started up as a garage operation in Torrance, CA, about 15 miles south of Copeland’s house at the time. Day Runner was producing many of the same things—forms for household organization, expenses and so forth—but in the thousands. Today, it’s one of the major players in this market.)

### The Page Macros

Let’s begin by talking about the page macros themselves. The stock pages are 6.75-inches tall by 3.75-inches wide. For each page, we assume the left page edge corresponds to the `troff` page offset, contained in the `.o` number register. Within the page, the text is indented by the `troff` indent in the `.i` register. Setting up our code with as many parameters as we can, we’ll store the full page width in the `w` register, its length in the `L` register and the usable page width—which allows for left and right margins—in the `w` register. Given those variables as our starting point, we need a macro to provide the cut marks at the corners of the page:

```
.\ " cut mark: horiz-pos
.\ " [orientation: {u|d}]
.de CM
.sp |\\$1
.nr t \\n(cmu)
.nr u 2u*\\n(cmu)/3u
.nr v \\n(cmu)/3u
```

# Work

```
.nr x \n(.o
.nr y \n(.i
.nr z \n(.s
.ps
.po
.in
.ps 5
.po \nxu-\ntu
.in 0
.lt \nWu+\ntu+\ntu
.ie "\\$2"u" .tl @\l'\n\nuu'\h'\nvu\
  '\v'-\ntu'\L'\n\nuu'\v'\nvu@\
  @\v'-\ntu'\L'\n\nuu'\v'\nvu\
  '\h'\nvu'\l'\n\nuu'@
.el .ie "\\$2"d" .tl @\l'\n\nuu'\h'\nvu'\
  \v'\ntu'\L'-\n\nuu'\v'-\nvu@\
  @\v'\ntu'\L'-\n\nuu'\v'-\nvu'\
  \h'\nvu'\l'\n\nuu'@
.el .tl @\l'\n\nuu'@\l'\n\nuu'@
.ps
.ps \nz
.po
.po \nxu
.in
.in \nyu
.lt
..
```

We provide the width of the cut mark as a parameter in the `cm` register, which is only used by this macro. We do a great deal of hand-waving to carefully save and restore the page offset and indent. We'll need to put the left cut mark not only to the left of the current indent, but to the left of the logical page boundary as well. Notice that we try to honor `troff`'s one-deep stack of indents and page offsets by saving and then restoring the previous values. The guts are in the `if-else` sequence. This sets the cut mark as a title line. In general, we'll use this macro by providing an upward-pointing cut mark at the top of the page and then a downward-pointing one `\nLu` further down:

```
.CM li u
.CM li+\nLu d
```

If you look ahead to Figure 3 on Page 47, you'll see cut marks in all four corners of the sample page.

Next, we want to show where the holes should be punched.

```
\n six holes at page offset: top-of-page-y
.de HO
.sp |\n\n$1
.nr x \n(.iu
.in
.in 0
.sp .875i
\h'.4c'\D'c .4c'
.sp .75i-1v
\h'.4c'\D'c .4c'
```

```
.sp .75i-1v
\h'.4c'\D'c .4c'
.sp 2i-1v
\h'.4c'\D'c .4c'
.sp .75i-1v
\h'.4c'\D'c .4c'
.sp .75i-1v
\h'.4c'\D'c .4c'
.in
.in \nxu
.sp |\n\n$1
..
```

Given the top-of-page position as an argument, and having the page offset at the left edge of the sheet, positioning the circles for the holes is a simple business. Again, the only complication is saving and restoring the current and previous page indent.

In addition, because we began this project before duplex printers were common, we have a macro to generate registration marks on the page to ensure that we aren't sloppy when reinserting the paper into the printer.

```
.\n registration mark: xpos ypos [dia]
.de RG
.nr u \n\n$1
.nr v \n\n$2
.nr x \n(.o
.nr y \n(.i
.po
.in
.po \nOu
.in 0
.br
.ie \w"\n\n$3" .nr z \n\n$3/2u
.el .nr z .li
.sp |\n\nv
\h'\n\nuu'\h'-\n\nzu'\D'c 2u*\n\nzu\
  '\h'-3u*\n\nzu'\D'1 4u*\n\nzu 0\
  '\h'-2u*\n\nzu'\v'-2u*\n\nzu\
  '\D'1 0 4u*\n\nzu'
.sp -1v
.if "\n\n*S"f" \h'\n\nuu'\v'-\n\nzu'\h'-\n\nzu\
  '\h'-\w'\s6FRONT'u'\s6FRONT\s0
.if "\n\n*S"b" \h'\n\nuu'\v'-\n\nzu'\h'-\n\nzu\
  '\h'-\w'\s6BACK'u'\s6BACK\s0
.in
.in \nyu
.po
.po \nxu
..
```

We invoke this macro with the horizontal and vertical positions of the registration mark's center. It's positioned horizontally with respect to the base page offset in the `o` register. It simply draws a circle of the specified diameter, with crosshairs overlaid. If we have defined the string register `s`, it allows us to specify whether we're drawing a mark on the front or back of the page so that we

Figure 1. Registration Marks



label the cut mark. Figure 1 shows examples of the registration marks. Normally, we'd use the registration mark on each side of a physical page, but we need to be careful to position it in the same relative horizontal place on both sides of the paper.

Given that setup, we can provide a macro for generating the setup for a logical page on the front or back of one sheet of physical paper:

```
.de NP
.\" new sheet: xpos ypos {f|b} indent linelength
.po \\$1+\\nOu
.CM \\$2 u
.CM \\$2+\\nLu d
.ie "\\$3"b" .ds S b
.el .ie "\\$3"v" .ds S b
.el .ds S f
.nr I \\$4
.in \\$4
.nr w \\$5
.nr X \\$1
.nr Y \\$2
.sp |\\$2
..
```

This macro allows us to specify the upper left corner of the logical page, setting the page offset to the left edge. It also saves information on whether we're producing a recto or verso image, defaulting to the front side.

Last, it would be nice to have a macro that allows us to mount fonts in the first three positions. This will allow us (in good troff style) to refer to fonts by position, rather than name. The following macro is overkill for doing this:

```
.\" ----- font positions
.ds i R \" last mounted
.ds j I
.ds k B
.ds f R \" currently mounted
.ds g I
.ds h B
.de FP
.ie !\"\\$1\\$2\\$3" \\{
.\" we had args: mount \\$1 in pos 1, etc.
.ds i \\*f\" save current
.ds j \\*g
.ds k \\*h
.if !\"\\$1" \\{ .fp 1 \\$1
.ds f \\$1 \\}
.if !\"\\$2" \\{ .fp 2 \\$2
.ds g \\$2 \\}
```

```
.if !\"\\$3" \\{ .fp 3 \\$3
.ds h \\$3 \\}
.\\}
.el \\{\\}
.\" no args: flip (f,g,h) and (i,j,k)
.ds x \\*i \" pos 1
.ds i \\*f
.ds f \\*x
.ds x \\*j \" pos 2
.ds j \\*g
.ds g \\*x
.ds x \\*k \" pos 3
.ds k \\*h
.ds h \\*x
.\\}
..
```

This maintains a one-deep stack of fonts so that we can flip back and forth between two font families, which is probably not necessary.

## Providing Content

At this point, we have the infrastructure we need to begin putting together pages, but we need some macros to help us put information on those pages:

```
.de LI\" line: mark
.ie "\\*S"b" \
|h'-\\nBu'\\kx\\$1|h'|\\nxu'\\v'.25m\\
'D'1 \\nwu+\\nIu-\\n(.iu+\\nBu 0\\
'\\v'-.25m'
.el \\kx\\$1|h'|\\nxu'\\v'.25m'D'1 \
\\nwu+\\nIu-\\n(.iu+\\nBu 0'\\v'-.25m'
.br
..
.de ML\" multiple lines: count mark
.LI \\$2
.nr x \\$1-1
.if \\nx .ML \\nx \\$2
..
.de DL\" double lines
.LI
.sp -1v
.sp .2v
.LI
..
```

The first gives us a label with a line under it so that we can do something like: `.LI 8am`. The second produces multiple lines with the same label, such as a set of lines with little boxes at the left side: `.ML 10 \\(sq`. The third will generate a double line as a separator. We've used all three of these macros in the example shown in Figure 3 on Page 47.

Given all that, we can build arbitrary tables for our forms:

```
.nf
.na
```

```
.di TB
.ll \nwu
\!.ll \nwu
.ps -1
.TS
box tab(~);
c s s s
l | c | c | c .
Daily Exercise
=
weights~lbs~sets~reps
-
\0\0curl~
-
\0\0press~
-
\0\0legs~
=
running~
.TE
.ps +1
.di
```

This actually builds a diversion containing the table. Notice that we've specified the line length twice: once when we read the diversion and the second time is protected by the \! to be read when we replay the diversion. In effect, we now have a macro to display an exercise chart, which we can use anywhere, for example, in the code we used to generate Figure 2.

What are we missing? We only have the tools to build forms in portrait orientation; we don't have hooks in our macros to build landscape forms. A book list, for example, would be more natural laid out horizontally, rather than vertically. A check register form also comes to mind as one that's more convenient in landscape orientation. We'll leave this as our first exercise for the reader.

While we can build data sheets—we could show you a chart of aerobic points vs. time and distance on the treadmill—we don't have the bottom-of-page processing to allow us to generate arbitrary text as successive pages. We'll leave that as an exercise, too.

Last, two of our logical pages fit with room to spare on a single letter-size page, or even a slightly narrower European A4 page. We have a few macros to handle this, but as a third exercise, can you come up with a version of your own? Even

better—and we don't have a solution to this one—can you combine the previous two exercises by building the diversion hooks so that we can prepare running text input linearly, and then have logical pages one and three printed on the front of the physical sheet and logical pages two and four on the rear?

## The Daily Pages

As desktop and workgroup laser printers became affordable, it became clear that the next step was to provide custom daily calendar pages. The critical question was: "How come nobody does a day planner with the phases of the moon?"

Let's approach this sideways and start with a (relatively) simple page layout:

```
.de PG" basic daily calendar page
.FP H HO HB
.HO \nYu
.sp |\nYu
.sp .325i
.lt \nwu
.tl ""\ $1"
.sp -.75v
.DL
.sp -1v
.mk *x
\h'2.25i+3p'\v'.25m'\D'1 0 \
|\nYu+\nLu'\v'-.25m'
.sp |\n(*xu+1v
.nr *y \nwu
.nr w 2.25i
\|\(sq\|\|\To Do\(\emWork:
.sp -1v
.LI
.ML 19 \h'1m'\(sq
.sp -1v
.sp 2p
.LI
\|\(sq\|\|\To Do\(\emHome:
.sp -1v
.LI
.ML 16 \h'1m'\(sq
.in +2.25i+1n
.sp |\n(*xu-1v
.vs .5i
\08:00
\09:00
10:00
11:00
12:00
\01:00
\02:00
\03:00
\04:00
\05:00
evening
.nr w \n(*yu
.in
```

Figure 2. Exercise Chart

Daily Exercise			
weights	lbs	sets	reps
curl			
press			
legs			
running			

# Work

.vs  
..

This presupposes that we've already invoked the `.NP` macro we developed earlier and have the page cut marks and positioning set. We do some initial setup for the page, including putting the macro's argument in the upper right corner. Then it's simply a matter of drawing some lines with check boxes (using the `.ML` macro from earlier) for our "at work" and "at home" to-do lists. To the right of those lists, we provide a place to write appoint-

ments. Given that, we now have a page on which to hang the moon. And if you look at Figure 3, you'll see that's just what we've done. But how did the moon get there?

Calculating the phase of the moon is just a matter of crunching numbers, and we'll share a routine we've found for doing just that on our Web site. More interesting is how we produce a picture of that partial moon.

The moon is new—that is, completely dark—when its angle is zero relative to the line from Earth to sun. It is full when it is on the opposite side of the Earth from the sun and its angle is 180

degrees. If we have the phase as an angle,  $\phi$ , we can use the `troff` arc-drawing functions to generate a picture. That picture will generally be two arcs, the regular outer edge of the moon and the curve made by the Earth's shadow. The radius of the outer edge arc is the radius of the full moon; let's call it  $R$ . Then, the radius of the other arc is  $R * \tan(\phi)$ . Because the `troff` arc drawing function draws an arc from the current position  $(x,y)$  to  $(x+dh1+dh2,y+dv1+dv2)$  with a center at  $(x+dh1,y+dh2)$ , and with the knowledge that the endpoints of the two arcs must meet, it is merely a matter of algebra to figure out what the arc-drawing directives for the moon are. Further, because the quadrants of the moon are similar, once we have the proper calculation for the first quadrant, generating the arcs for the other three is merely a matter of flipping signs (see Listing 1, Page 48).

This subroutine draws the outer arc of a waxing or waning moon, and then, within those branches of the `if`, it handles either a concave or convex arc for the shadowed side. Full and new moons are handled as special cases, though our display of the new moon depends on a PostScript printer and, in particular, the relative size of the bullet from the PostScript special character font. For example, for a moon phase of 248 degrees, the routine above produces

```
\v'-20p'\D'a 0 10p 0 10p'\D'a \
-16.00p -10p 16.00p -10p'
```

which generates Figure 4.

We now have (in outline at least) all the tools we need to build arbitrary pages for our daily planners. Of course, if you want to use the phase of the moon code, you'll need to build a scaffold around it to generate the moon pictures interspersed with calls to a macro that

Figure 3. Planner with Phases of the Moon

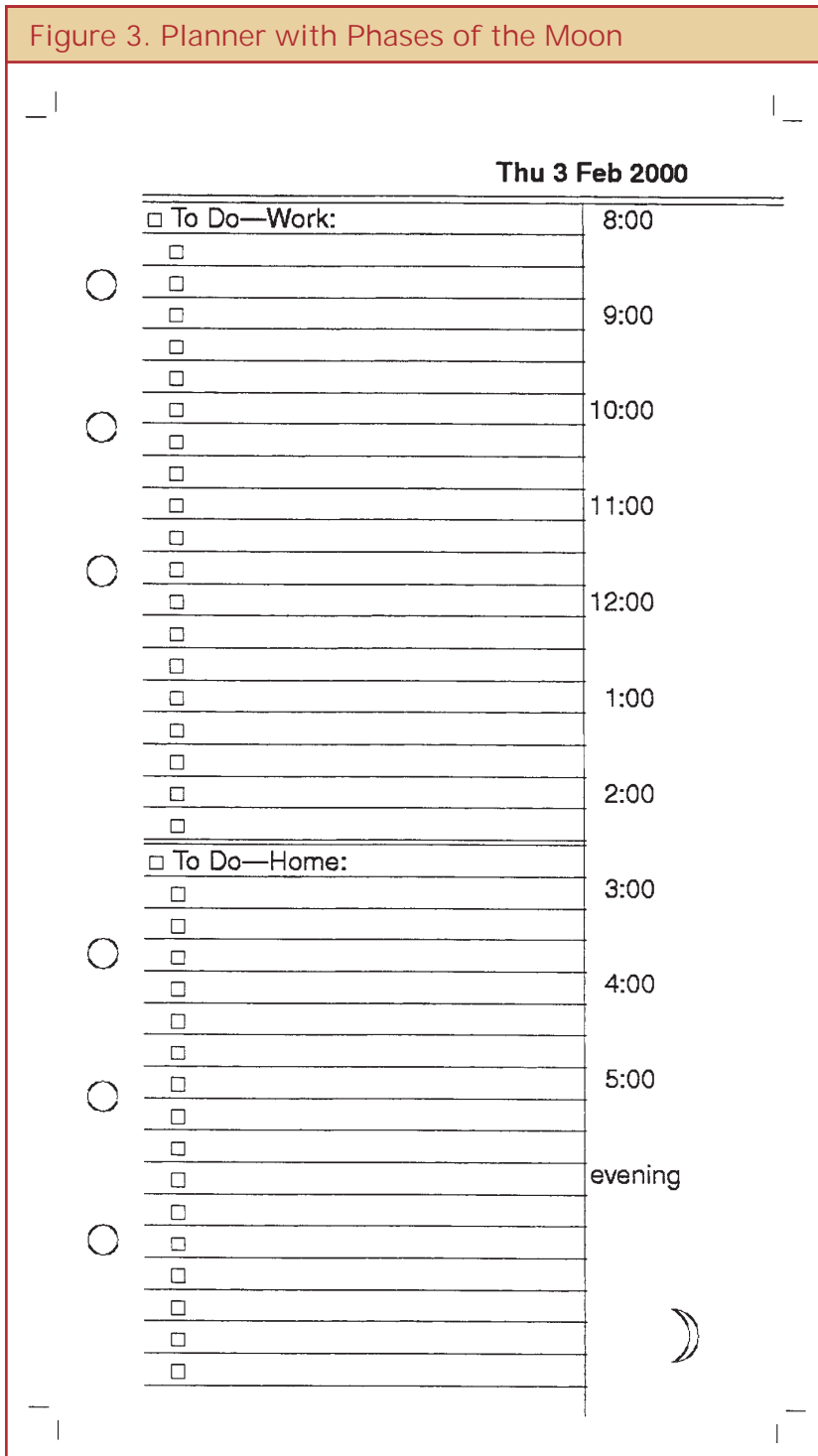


Figure 4. The Moon



## Listing 1. Drawing the Phase of the Moon

```

/* this draws a picture of the phase of the moon */
static char *id = "$Id: ...";
#include <math.h>
#ifdef M_PI
#define M_PI 3.1415929535897932384626433
#endif

moon_pic ( phase )
int phase;
{
    int radius = 10; /* moon radius in points */
    float phi;

    /* We print only the troff commands to
    produce the moon picture. We assume
    the caller produces newlines before and
    after, as appropriate, and ensures we
    are in the correct position. */

    /* new moon is a special case */
    /* we use the bullet on the PostScript
    special font; if we aren't printing
    on a PS printer, this code must change */
    if( phase == 0 )
    {
        printf( "\\s(%02d\\v'.05m'\\h'-\\w'",
            radius * 5 );
        printf( "\\(bu'u/2u'\\(bu\\s0");
        return;
    }

    /* handle anything but a new moon */
    phi = tan(phase*2.*M_PI/360.) * radius;

    if( phase < 180 )
    {
        /* the semi-circle on the right */
        printf( "\\D'a 0 %dp 0 %dp'",
            -radius, -radius );
        /* the other part of the crescent */
        if( phase < 90 )
            printf( "\\v'%dp'", 2*radius );
        printf( "\\D'a %.2fp %dp %.2fp %dp'",
            -phi, -radius, phi, -radius );
        else if( phase > 90 )
            printf( "\\D'a %.2fp %dp %.2fp %dp'",
                -phi, radius, phi, radius );
        else /* exactly 90, so we get a half-circle */
            printf( "\\D'l 0 %dp'", 2*radius );
    } else if( phase > 180 )
    {
        /* a semi-circle on the left */
        printf( "\\v'-%dp\\D'a 0 %dp 0 %dp'",
            2*radius, radius, radius );
        /* the other part of the crescent */
        if( phase < 270 )
            printf( "\\D'a %.2fp %dp %.2fp %dp'",
                -phi, -radius, phi, -radius );
        else if( phase > 270 )
            printf( "\\v'%dp'\\D'a %.2fp %dp %.2fp %dp'",
                -2*radius, -phi, radius, phi, radius );
        else /* exactly 270: we get a half-circle */
            printf( "\\D'l 0 %dp'", -2*radius );
    } else
        /* phase is 180 --> full moon */
        printf( "\\v'%dp'\\h'%dp'\\D'c %dp'",
            -radius, -radius, 2*radius );
    }
}

```

generates the daily page. The full macros and code we've discussed are available in the usual place, <http://alumni.caltech.edu/~copeland/work/>, and we encourage you to add to them and share your additions. But now let's return to an item we discussed a while back.

## Electronic Books Revisited

A year ago, we issued a challenge (see "Reader, Part 2," January 1999, Page 40, <http://sw.expert.com/C9/SE.C9.JAN.99.pdf>). Electronic book hardware had just become available from NuvoMedia Inc. and SoftBook Press, and we suggested that these companies should provide reader software so their electronic books could be read on general-purpose computers. In particular, we noted, "We already carry a lot of hardware when we travel—our normal mode of operation is to clear an airport with more weight in our briefcase than in our suitcase. Just thinking about adding an extra two pounds for the specialized hardware makes our shoulders hurt."

NuvoMedia is now giving away a software package called eRocket, which provides the look-and-feel of its custom e-book hardware on your screen. (Pick up your own copy at <http://www.rocket-ebook.com>.) Unfortunately, the company provides a binary-only distribution and only provides executables for Windows systems. Also, because the software-only package doesn't have the ability to interact with a separate piece of hardware at your end of the download, you can only read the free editions. That is, because there is no secure way of emulating the security hardware built into the Rocket hardware and its cradle, you can't yet buy and download the latest (encrypted) Thomas Harris blockbuster from [barnesandnoble.com](http://barnesandnoble.com).

After spending an evening reacquainting ourselves with *Alice's Adventures in Wonderland* using eRocket, we can certainly understand the advantages of having a custom piece of hardware for leisure reading, particularly one that's smaller, lighter and easier to use in bed than our laptop.

The next interesting step would be for NuvoMedia to provide a version of the eRocket software for the Palm Pilot or other generic portable hardware.

Enough of this speculation. We hope that the last year of the millennium started nicely for you. We will return with another interesting problem next month. In the meantime, go forth and make interesting calendar pages.

Until then, happy trails. ✍

---

*Jeffrey Copeland* ([copeland@alumni.caltech.edu](mailto:copeland@alumni.caltech.edu)) is currently living in the Pacific Northwest, where he spends his time writing UNIX software in a large development organization and fighting damp rot.

*Jeffrey S. Haemer* ([jsh@usenix.org](mailto:jsh@usenix.org)) works at QMS Inc. in Boulder, CO, building laser printer firmware. Before he worked for QMS, he operated his own consulting firm and did a lot of other things, like everyone else in the software industry.

Note: The software from this and past Work columns is available at <http://alumni.caltech.edu/~copeland/work> or alternately at <ftp://ftp.expert.com/pub/Work>.