On the issue of whether Lyrix has most of the capabilities of vi, I think you and I differ on the semantics of the word capabilities. *Lyrix does have most of the capabilities of vi, even though vi is a text editor.*

Your comments on the customization features of Lyrix are correct. I have used the customization features to a much greater extent than I mentioned in the review. This feature is one of Lyrix's greatest selling points.

I did verify several of the problems that you pointed out, which I had not picked up in my usage. In regard to Lyrix's speed on my IBM PC, the occasional response problems also appear with vi on my PC, so I don't think that Lyrix itself has a significant speed problem. I may be a little biased, because the time-sharing systems that I use in my work are extremely slow due to the large number of users. Lyrix on my PC is faster than my systems at work by a large factor.

—George R. Allen

C, More

In the June C interpreter review, Mr. Unger didn't point out the most obvious advantage to using the C-terp interpreter. When you set up C-terp with your current compiler, C-terp offers exactly the same functions and features as the compiler. This means that you don't have to create two versions of a program, one for C-terp and one for the compiler.

By using C-terp, I can write a 10,000-line program using all the functions of the Microsoft compiler and still run it under the interpreter to find a bug or error. When you're dealing with large programs with long compile times, C-terp is a godsend. If I had to take my 10,000-line program (in 20 to 30 files) and, for example, change all occurrences of `get( )` to `getline( )`, I could never get a program developed.

C-terp is too expensive if you are only trying to learn C, but it is well worth the money for professional programmers. If you develop serious C programs, once you try C-terp you will never go back.

P. Lyle Mariam
St. Louis, MO

I pointed out in my review that if you have a copy of one of the five C compilers supported by C-terp, you can create a version of the interpreter that uses all the functions that are available with that compiler.

In fact, you can add other library functions to the interpreter using a simple but somewhat tedious procedure; I added the entire Essential Graphics graphics functions to a version of C-terp. It increases the size of the interpreter program and

continued

consequently decreases the size of the source code you can use.

When used in this way, C-terp is a tremendous time saver for developing programs that you will later compile.

—John Unger

C Syntax Checker

I read John Unger's review of C interpreters (June) with great interest. C interpreters are created as program-development and debugging environments, not as compilers, and the error messages they generate are one of the most important issues to consider when selecting one. Mr. Unger deserves particular commendation for addressing this concern.

As a user of the Microsoft C compiler, I spent hours trying to locate mysterious errors before I realized that the most common mistakes are simple typos, which the compiler usually notices several lines past the place where they actually occur. Frustrated with missing braces and other details, I wrote a program (see listing 1) that performs a quick and rudimentary syntax check on my source code. The exit codes allow the use of this syntax checker in batch files, making the entire compilation even easier.

Jerzy Tomasik
Long Beach, CA

Listing 1: *Syntax checker for C source code.*

```
/* syntax.c Program running a quick
   syntax check on C source code
   Version 1.11
   J. Tomasik; created 05/23/87
*/
#include<stdio.h>

main( argc, argv )
int argc;
char *argv[];
{
   FILE *fopen( ), *infile;
   char c;
   int lbrace = 0, rbrace = 0,
   squote = 0, dquote = 0, lpar = 0,
   rpar = 0; int rbrkt = 0, lbrkt = 0;
   int bytecount = 0, errorcount = 0;

   if( argc != 2 ) {
      printf( "SYNTAX checker for C
       source code, version 1.1\n
       Copyright (C) J. Tomasik 1987,
       1988\n\n" );
      printf( "Usage: syntax
       fname.ext\n" );
      exit(1);
}
infile = fopen( argv[1], "r" );
if( infile == NULL ) {
      printf( "Cannot open %s\n",
       argv[1] );
      exit(2);
}

while( (c=fgetc(infile)) != EOF ) {
      ++bytecount;
      if( c == '{' )
        ++lbrace;
      else if( c == '}' )
        ++rbrace;
      else if( c == '\'' )
        ++squote;
      else if( c == '\"' )
        ++dquote;
      else if( c == '(' )
        ++lpar;
      else if( c == ')' )
        ++rpar;
      else if( c == '[' )
        ++lbrkt;
      else if( c == ']' )
        ++rbrkt;
}
fclose( infile );
printf( "The file length is %d\n",
    bytecount );
if( lbrace != rbrace ) {
      printf( "There are %3d left and
       %3d right braces\n", lbrace,
       rbrace );
      ++errorcount;
}
if( lpar != rpar ) {
      printf( "There are %3d left and
       %3d right parentheses\n",
       lpar, rpar );
      ++errorcount;
}
if( lbrkt != rbrkt ) {
      printf( "There are %3d left and
       %3d right brackets\n",
       lbrkt, rbrkt );
      ++errorcount;
}
if( squote % 2 ) {
      printf( "The single quote marks
       are not paired\n" );
      ++errorcount;
}
if( dquote %2 ) {
      printf( "The double quote marks
       are not paired\n" );
      ++errorcount;
}
if( errorcount == 0 ) {
      printf( "No errors found, OK to
       compile\n" );
      exit(0);
}
else
      exit( errorcount );
}
```

continued

Understanding C

In the review of C interpreters in the June BYTE ("Four C Language Interpreters" by John Unger), Mr. Unger lists a single "major shortcoming" of C-terp as "its lack of a built-in library." What is horrifying about this statement is that it shows a fundamental failure to understand the reason for the product's existence.

C-terp is for those who need to develop code for their compilers. It requires you to use the compiler's libraries, because it is trying to limit the degree to which the same source causes different results when run under the interpreter and when compiled by the compiler. In short, C-terp is designed to be a development environment. Indeed, you cannot order C-terp without specifying which compiler you will be using. To fail to understand that you must use your own compiler's library of routines with C-terp is to fail to understand the nature of the product. It's much like complaining that a calculator is a flawed product because it's too mathematical.

There also seems to be a notion that C interpreters are a great way to learn C. Certainly Gimpel Software is not perpetuating that misconception, but the idea exists nonetheless. BASIC is an ideal learner's interpreted language, because the significant unit in BASIC is the line. It is not idiotic to sit down at a computer keyboard and start writing BASIC to learn it. However, to produce code in a modular language you must understand the structure of the language, the scope of variables, and so on. In C, the words of the language are less important than the structure. Avoiding syntax errors is not the heart of learning C.

At the Eye Research Institute, we recently purchased a site license for C-terp because it is so useful. In particular, it has a good line editor, lets you run quick checks to make sure you didn't leave off a semicolon, and lets you quickly test what actually comes out of a function before developing too much code for easy debugging. Furthermore, C-terp is a dream for, say, graphics-routine development. With C-terp, you can interactively develop what you want to see. In such an application, it's a minor miracle to have your first compiled output the only compiled output.

Tom Clune
Boston, MA

First, let me add the next three words of the sentence that you quoted from my review. The entire phrase is, "its lack of a built-in library of mathematical functions." C-terp comes with a complete built-in library of extremely useful functions and is lacking only in this one specific area—support for math functions. Because both Run/C and Instant-C include mathematical functions in their built-in libraries, I thought it was fair to point out the omission of such functions in C-terp's.

C-terp is designed primarily for use in a development environment as a companion program to a specific C language compiler; this makes its own lack of math functions not as crucial. But C-terp can also be used alone, as a tool to learn C, to test concepts of the language, and to produce useful programs that can run within the confines of its interpreter environment. It is wrong to imply that C-terp must be used with a specific compiler. I agree, however, that any C language interpreter is most useful when it can work with source code that can be seamlessly ported between the interpreter and compiler environments.

I wholeheartedly support your opinion that trying to learn C using a mindset developed in BASIC is a serious mistake.

continued

However, an interpreter's ability to catch syntax errors quickly, which you yourself mention, does make it a useful learning tool for beginning C programmers.

—John Unger

Alternate Approach to DTP

The theme of the May issue of BYTE was desktop publishing; however, nowhere in the articles or charts was there any mention of the PowerText Formatter, an $89.95 desktop-publishing product announced and shown at PC-Expo in July 1986.

John W. Seybold's view of desktop publishing is but one approach; he dismisses all approaches other than WYSIWYG. But current Macintosh and IBM WYSIWYG software leaves a lot to be desired and suffers from some fundamental problems, and the alternatives may be more cost-effective, both in initial cost and in day-to-day operation in a production environment.

WYSIWYG is really only approximately what you get. The fonts differ from screen to page, and interletter and word spacing differ. What looks nice kerned on the screen can often end up as touching characters on the printout, and what seems to be centered on the screen may not be when it's printed.

Scaled fonts, such as those of PostScript, do not map onto dots very well. In addition, a good typographer will often change the shapes of letters in different sizes simply because they look better. Mathematics can't do this. As a result, scaled fonts are not as crisp and clear as fonts discretely designed for each point size.

WYSIWYG systems don't function very well in environments where several people supply the copy and where external artwork and halftones have to be factored in. And, at least on the IBM PC and the Macintosh, using WYSIWYG screens to lay out metro-size newspapers is somewhat like painting through a keyhole.

WYSIWYG requires a lot of hardware. In the PC arena, one really needs a PC AT–class machine, an EGA card, a hard disk drive, and a mouse, not to mention the laser printer. Can everyone who needs desktop publishing really afford all this hardware?

WYSIWYG is ideal for flyers and short newsletters. But is it really practical for books of 200 or more pages?

When you strip away the hype from desktop publishing, what you really find is a problem of economics. Typesetting costs a lot of money. That problem can be addressed by 300-dot-per-inch (dpi) laser printers, at least in typesetting textual material. Page-layout and composition programs have their place, but they are only a part of the typesetting and publishing problem. When the visual aspects of each individual page are as important as the textual aspects, then page-layout programs may be the ultimate solution. However, advertising material, flyers, and newsletters represent only a very small percentage of the printed material produced in this world. Are we to believe that the economic solution to the high cost of typesetting for each and every page printed is to sit in front of a screen with a mouse?

Your theme articles indicated that desktop-publishing hardware costs between $10,000 and $15,000, with software running between $200 and $800. But consider this: The street price of an HP LaserJet Series II printer is about $1700, a good set of times roman and helvetica fonts costs about $155, and the PowerText Formatter costs $89.95. Inset 2, a graphics-capturing and editing program from American Programmers Guild Ltd., costs $99, and a clone costs

continued