Some Macros to Draw Crosswords*

B Hamilton Kelly†
Extensions by Dirk Laurie

February 1, 2010

Abstract

The crossword environment is intended to be used to typeset crossword puzzles for use in newsletters, etc.

Contents

List of Figures

*This file is v3.0, dated 2010/02/01

†Especial thanks to my colleague Niel Kempson for many helpful suggestions, and to Frank Mittelbach of the Johannes Gutenberg University of Mainz, who saved me two pages of code!

1 Introduction

As a small diversion from the statistics of computer availability, lists of new software, and the like, Computer Centre Newsletters often include a crossword for the amusement of their readers.[1]

The macros presented in this document provide a LaTeX method of typesetting these, and also assist the composer to ensure that the "grid" all goes together correctly. That is to say, checks are made that crossing lights have the same letter where they cross, lights starting at the same square have the same clue number, and clue numbers are in sequence from 1 upwards when reading from top left to bottom right.

The grid generated can be the more usual form, with black squares separating the "lights" which receive the answers to the clues, or the *Mephisto/Azed* type of grid, in which only thicker grid lines separate the lights, or even a mixture of these. The user need not specify which type of grid is to be used, since they are both described by the same simple rule: a bar is drawn between every pair of adjacent light squares that do not belong to the same light.

A sample crossword appears as Figure 1; I've left the grid blank for those who want some intellectual exercise: those who don't can cheat by reading the source listing at the end of this article!

The whole crossword, including the `\clue` commands (*q.v.*), is bracketed within the crossword environment. The user may specify one optional and must specify two required parameters:

⟨***gridrows***⟩ The number of rows in the rectangular grid. If supplied, it must be enclosed in square brackets. It may be omitted if the number of rows equals the number of columns.

⟨***gridcols***⟩ The number of columns in the rectangular grid.

⟨***visible***⟩ This controls whether the answers are to be "filled in"; obviously of no use for publication, but useful whilst composing the crossword. If the parameter provided is the letter '`Y`', then the answers will be typeset; if '`N`' then the lights will be left blank. Any other value[2] provided for this parameter will cause LaTeX to input a yes/no answer by interaction with the user.

An analogous environment is provided especially for typesetting a smaller version of a grid showing, for example, "Last Month's Solution". In this of course, the answers *always* appear, and the clues are not printed. Again it takes three parameters, the first optional:

⟨***gridrows, gridcols***⟩ As before, these specify the number of squares in the two axes.

⟨***header_text***⟩ Some text which will be set (in **bold**) above the completed grid.

Here is an example of the crossword* environment:

[1] That at RMCS has a bottle of wine as a prize!

[2] The lower-case letters '`y`' and '`n`' are also recognized

Last month's solution

Within the body of these environments appear a succession of `\clue` commands; each of these takes a total of seven (!) parameters:

⟨***clue_number***⟩ The number of the light on the grid, for example `{17}`. See 1.2 below for details of how more complex specifications may be given for multiple lights.

⟨***Across/Down***⟩ This parameter *must* be either the letter '`A`' or '`D`', in upper-case.

⟨***col_number***⟩ The x-coordinate of the first square of the light. The left-most column of the grid is numbered 1.

⟨***row_number***⟩ The y-coordinate of the first square of the light. The top-most row is numbered 1.

⟨***answer***⟩ The answer to the clue (or that part of it which appears in the light numbered ⟨*clue_number*⟩). Traditionally in a completed grid, this is a string of upper-case letters, but spaces in between are silently ignored, lower case letters are equally OK, and some other characters such as question mark and fullstop are useful as placeholders in an uncompleted grid. No hyphens or apostrophes, though.

⟨***text***⟩ The text of the clue itself. If you want to use any LaTeX macros in this text, such as `\dots`, each such macro must be preceded by `\noexpand`. This includes such macros as `\&`, to produce an explicit ampersand.

⟨***help***⟩ Anything to appear after the text, in parentheses; this will most usually be used for giving the length of the answer, such as "7" or "2,6,3-3". Also used when the text of the clue is associated with another light when this parameter may say something like "see 14d". If this parameter is left *absolutely empty*, not merely blank, i.e. `{}`, the clue (if any) will not be printed at all.

1.1 Customizing the layout

Some definitions made when entering the crossword environment may be overridden by the user inside the environment.

```
\unitlength 6mm
\def\Preamble{}
\def\cluewidth{70mm}
\let\numbersize\tiny
\let\cluesize\ninept
\def\barwidth{0.1}
```

These refer respectively to size of grid cells, text to be set between the diagram and the clues, width of clue columns, typesize of numbers in grid,

type size of clues, and width of bars in `\unitlength`s.

If you override `\cluesize` by a LaTeX size command like `\normalsize`, you will probably feel that there is too much vertical space between the clues. The only good solution is to write your own macro by adapting the definition of `\ninept` given below.

1.2 How to Specify Clue Numbers

Sometimes the solution to one clue is split amongst a number of lights. To cover this eventuality, provide a `\clue` for *each* of the lights involved, with the solution to that light *alone* given as ⟨*answer*⟩. All except the `\clue` corresponding to the first light of the solution should have a null ⟨*text*⟩, and the ⟨*help*⟩ parameter might be something like "see 7d".

If you wish to omit any further reference to the clue number (the usual practice when the current clue applies to consecutive lights, or in hard cryptics), leave the ⟨*help*⟩ field totally empty.

The `\clue` for the first light of the solution should provide the *entire* clue as its ⟨*text*⟩, and the ⟨*help*⟩ might say something like "7,3-3". The ⟨*clue_number*⟩ field should consist of the number of that light, followed immediately by the text required to describe the other lights, separated from it by some non-digit character, for example, a space.

For example, suppose the clue "Bill's desired outcome?", has the solution 'ACT OF PARLIAMENT' which is to go into lights 9d and 13a. Then[3]

```
\clue{13}{A}{5}{1}{PARLIAMENT}{}{see 9d}
\clue{9 {\noexpand\rm\&} 13a}{D}{1}{10}{ACT OF}%
      {Bill's desired outcome?}{3,2,10}
```

will produce

13 (see 9d)

amongst the ACROSS clues, and

9 & **13a** Bill's desired outcome? (3,2,10)

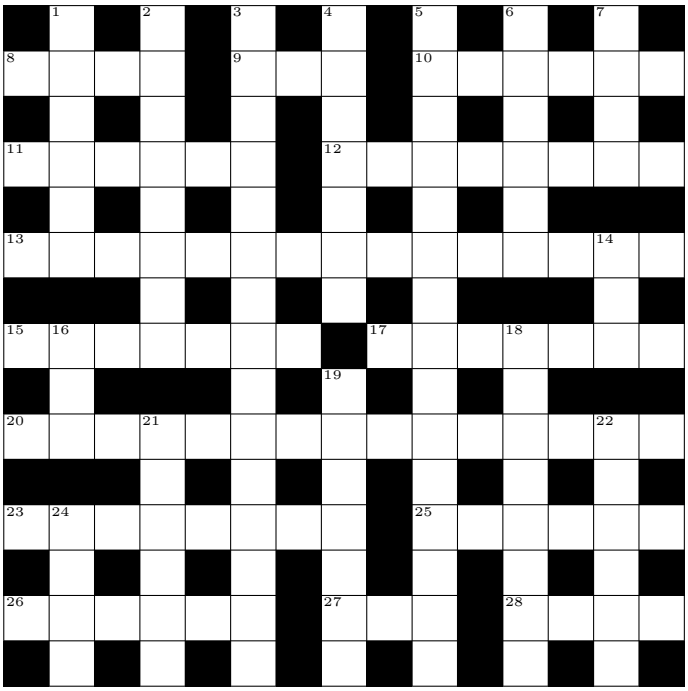
amongst the DOWN clues

2 Definition of the Macros

`\ninept` `\@listi` We define a new 9pt font size for setting clues. This command also defines suitable parameters for list environments set in this size of type.

```
1 ⟨*package⟩
2 \def\ninept{\@setsize\ninept{11pt}\ixpt\@ixpt
3   \abovedisplayskip 8.5pt plus 3pt minus 4pt
4   \belowdisplayskip \abovedisplayskip
```

[3] note the `\noexpand` before the `\rm` for the `\&`



ACROSS

8 Points a thousand tested for witchcraft. (4)
9 Gourmet's triumphant cry on finding middle-cut Pacific salmon! (3)
10 One hundred stride backwards across a Pole. (3-3)
11 Fifties' jazz record about Eastern childs' play. (2-4)
12 Timetable created by editor in synagogue of Spain. (8)
13 The dialect a girl mixed up tangle around symbolic diagram used by maritime student! (15)
15 Wire fastening bent road narrowly. (7)
17 Hammerhead consumes German company and casts a shade. (7)
20 Strange cel alien chops to make a figure with odd sides. (7,8)
23 Sounds like a hoarse editor came to in total! (8)
25 Assert without proof everyone, for example, English. (6)
26 Flourished examination of flowers. (6)
27 Floor covering discards fuming sulphuric acid and returns to nothing. (3)
28 "Latin for a candle" to be silent note about aircraftman? (4)

DOWN

1 Water rush noise disturbs show so! (6)
2 Mischievous child with cloth measure hesitates to assemble rotor. (8)
3 Mercifully inclined to pass round ten? Nay, about short blower! (15)
4 Sort ion? An isotron gives it a new twist! (7)
5 Wildly and boisterously rearrange Billy May Third, roughly. (15)
6 Satirical book or film—give odds about revision of "Dune"? (4-2)
7 & 24 Premier took in a Lord Lieutenant and all played an old game in London street. (4-4)
14 Work expended in power games? (3)
16 A church circle (or part of one). (3)
18 Encircle hindrances under hair in long curls. (8)
19 Boss over otorhinolaryngology department undergraduate. (7)
21 Old dovecote in Parisian museum. (6)
22 Like ornamental fabric, for example, that's in bequest. (6)
24 (See **7**)

Figure 1: A Sample Crossword

```
  \abovedisplayshortskip \z@ plus2pt
  \belowdisplayshortskip 4pt plus2pt minus 2pt
  \def\@listi{\itemsep 0pt
    \parsep \z@ plus 1pt
    \topsep 4pt plus 2pt minus 2pt
}}
```

2.1 Counters and Lengths

\ifnumberit
\numberittrue
\numberitfalse

The crossword environment draws a grid (with black and white squares); each light into which a clue's answer is to be written has to be numbered, and this number will be typeset (using \tiny) in the top-left corner of the first square of the light.

This style option also provides the crossword* environment, which is intended to be used to produce "last month's solution" in a smaller grid. There is insufficient room for clue numbers to appear on the grid in this mode, so \ifnumberit is used to indicate whether the numbers should be set.

```
\newif\ifnumberit
```

\gridr@ws
\gridc@ls
\p@rsize
\p@csize

The counters \gridr@ws, \gridc@ls, are used to hold the width of the grid, as the number of squares in each direction.

To prevent too much run-time arithmetic, the counters \p@rsize, \p@csize are set to be one count higher than \gridc@ls, \gridr@ws.

```
\newcount{\gridr@ws}\newcount{\gridc@ls}
\newcount{\p@rsize}\newcount{\p@csize}
```

\Down
\Across

As we move around the grid, determining whether squares are black or white, we utilize the counters \Across and \Down to keep track of our location.

```
\newcount{\Down}
\newcount{\Across}
```

2.2 Reading and Writing the Clues

\tf@acr
\tf@dwn
\OpenClueFiles

Whilst we are determining the appearance of the grid, we copy the text of each of the clues to an auxiliary file, so that the latter may later be read back to generate the clues themselves after the grid has been printed.

This macro opens a new file, with file extension .acr, and puts into it the commands necessary to typeset the Across clues. It also opens a .dwn file, which is similarly filled with the Down clues.

Next paragraph is no longer true. I changed the coding to allow more than 6 crosswords to be used in one document. Using the approach outlined below the system would run out of write channels pretty soon. (FMi)

[These files are created in the same manner as table-of-contents (.toc) files, etc; thus LaTeX will create file "handles" with names \tf@acr and \tf@dwn. However, that would ordinarily attempt to *read* the given file first, and also might defer the actual opening; therefore, we start a new group in which we redefine LaTeX's @input command and TeX's \openout primitive.]

```
\newwrite\tf@acr
\newwrite\tf@dwn
\def\OpenClueFiles{%
  \immediate\openout \tf@acr \jobname.acr\relax
  \immediate\openout \tf@dwn \jobname.dwn\relax
```

Here's the preliminary material that gets inserted into the `.acr` file.

```
  \@writefile{acr}{\begin{minipage}[t]{\cluewidth}}%
  \@writefile{acr}{ \centerline{\textbf{\ACROSStext}}}%
  \@writefile{acr}{ \sloppy}%
  \@writefile{acr}{ \cluesize}%
  \@writefile{acr}{ \begin{ClueList}}%
```

Whilst something similar goes into the `.dwn` file.

```
  \@writefile{dwn}{\begin{minipage}[t]{\cluewidth}}%
  \@writefile{dwn}{ \centerline{\textbf{\DOWNtext}}}%
  \@writefile{dwn}{ \sloppy}%
  \@writefile{dwn}{ \cluesize}%
  \@writefile{dwn}{ \begin{ClueList}}}
```

`\ACROSStext` `\DOWNtext` You may (e.g. if the crossword is not in English) prefer to change the column text headers.

```
\newcommand\ACROSStext{ACROSS}
\newcommand\DOWNtext{DOWN}
```

`\CloseClueFiles` After the grid has been printed, we can close the "clues" files; these will later be read back in (by the `\endcrossword` command) to set the text of the clues below the grid.

Before closing, we insert the material that completes the two ClueList environments; firstly across...

```
\def\CloseClueFiles{%
  \@writefile{acr}{ \end{ClueList}}%
  \@writefile{acr}{\end{minipage}}%
```

Then for the down clues.

```
  \@writefile{dwn}{ \end{ClueList}}%
  \@writefile{dwn}{\end{minipage}}%
```

Now we can close those files, and make them "invisible" if someone tries to write to them.

```
  \immediate\closeout\tf@acr \let\tf@acr\relax
  \immediate\closeout\tf@dwn \let\tf@dwn\relax
  \endgraf
}
```

2.3 Tabulating the Clues

The auxiliary files contain the texts of the clues, each given as an `\item` for the ClueList environment. This is similar to a description list, except that overlong labels run on into the text rather than sticking out to the left.

\ClueList
\ClueListLabel

This sets up the ClueList environment, and defines the appearance of the label.

```
\def\ClueListlabel#1{\hspace\labelsep {\bf #1}\hss}
\def\ClueList{\list{}{\labelwidth\leftmargin
  \advance \labelwidth by -\labelsep
  \let\makelabel\ClueListlabel}}
\let\endClueList\endlist
```

\PrintClues

The following macro reads in the two files (of Across and Down clues), and sets them alongside each other, separated by a vertical rule. Clues are set in the style of the ClueList environment.

```
\def\PrintClues{%
  \centerline{%
    \begin{tabular}{ c | c }
      \@input{\jobname.acr}
      &
      \@input{\jobname.dwn}
    \end{tabular}
  }\endgraf
}
```

3 Creating the Grid

The remaining commands are concerned with creating (and, optionally, populating) the crossword grid.

crossword
\c@ls

The crossword environment takes three parameters: *viz.* the number of rows (optional, may be omitted if the grid is square) and columns of the matrix, and the indication of whether the grid is to include the answers. (If the latter is not Y or N, LaTeX will request it interactively.) The macro \c@ls is a communication mechanism.

```
\newenvironment{crossword}[3][\c@ls]{%
    \def\c@ls{#2}
```

We start off with a \vtop box and a group to hold everything within the environment, so as to ensure that user-entered text remains with the crossword.

```
  \endgraf\leavevmode
  \vtop\bgroup
```

\Header
\cluewidth
\numbersize
\cluesize
\barwidth

The crossword environment uses the full-size grid, and has the lights numbered. Furthermore it doesn't have any heading to output (see the crossword* environment). Some other macros are given so that the user can customize the layout.

```
    \unitlength 6mm\numberittrue
    \def\Header{}%
    \def\Preamble{}%
    \def\cluewidth{70mm}
    \let\numbersize\tiny
    \let\cluesize\ninept
    \def\barwidth{0.1}
```

We now open the auxiliary files into which the clues are written, and determine (interactively if necessary) whether the answers are to be written into the grid.

```
    \OpenClueFiles
    \TestAnswers{#3}%
```

Finally, we generate the necessary macros to describe the grid. For each square, there are three macros. For example, corresponding to the square in row 4 from the top, column 2 from the left, the macros are:

`\LetterRivCii`	Letter in the square; `!` is used for a black square.
`\NumberRivCii`	Number in the square; 0 is used for a numberless square.
`\BarRivCii`	Code defining bars around the square, defined below.

```
    \SetUpGrid{#1}{\c@ls}}
```

When all the clues have been processed, we can invoke `\FinishGrid` to draw the grid. The `\FinishGrid` and `\PrintClues` commands draw the grid and tabulate the clues, respectively. By enclosing them in a vertical mode list, we ensure that they remain stuck together on one page!

The crossword environment defines `\Header` to be empty, but the user may give it an explicit definition within the environment; if so, we'll print it just above the grid itself.

```
{\endgraf
    \centerline{\Header}%
    \hbox{\FinishGrid}%
```

We can now finish off the auxiliary files and then read them back in to set the text of the clues below the grid.

Finally, we complete the group and the `\vtop` box.

```
    \ifthenelse {\equal{\Preamble}{}} {}
       {\centerline{\parbox{\textwidth}{\Preamble}}\medskip}
    \CloseClueFiles
    \hbox{\PrintClues}%
  \egroup
}
```

`crossword*` The crossword* environment doesn't need a second parameter to control printing of answers, because it **always** populates the grid with the answers. Instead, its second parameter provides the text to appear above the printed grid. Its actions are as for the crossword environment except that

- It assumes that there is descriptive text above the grid and allows room for it.
- It **always** outputs the answers, without numbers.
- It draws them in a smaller box.
- It doesn't output the clues (it doesn't even open any auxiliary files!)

```
\newenvironment{crossword*}[3][\c@ls]{%
  \def\c@ls{#2}
  \unitlength 4mm\numberitfalse
  \def\barwidth{0.1}
  \endgraf\leavevmode
  \vtop\bgroup
    \def\Header{{\bf\strut #3}}%
    \def\answer{Y}%
    \let\tf@dwn=\relax \let\tf@acr=\relax
    \SetUpGrid{#1}{#2}}
{\endgraf
    \centerline{\Header}%
    \hbox{\FinishGrid}%
  \egroup
}
```

3.1 Macros for each square

\ind@x
\arr@yref
\value
\assign
\addto

Crosswords are inherently two-dimensional, and require matrices of related definitions. Since TEX does not allow numbers in a name, each square has a unique text index using `R` and `C` followed by roman numerals. Each macro name consists of a prefix (the same for all squares) followed by the unique text index. The original version of this package had only one such macro for each square, with an empty prefix, but in this version we have three: `Letter`, `Number` and `Bar`.

For readability of the code, the following macros provide bracket syntax, thereby avoiding the use of `\csname`...`\endcsname` elsewhere. For example, `\value Bar[2,5]` gives `\BarRiiCv`. A new definition `NEW` is assigned by `\assign Bar[2,5]=NEW;`. If the contents of `\BarRiiCv` is a number, you can add 31 to it using `\addto Bar[2,5]+=31;`. In both cases, the semicolon is essential.

```
\def\ind@x#1#2{R\romannumeral #1C\romannumeral#2}
\def\arr@yref#1#2#3{\csname #1\ind@x{#2}{#3}\endcsname}
\def\value#1[#2,#3]{\arr@yref{#1}{#2}{#3}}
\def\assign#1[#2,#3]=#4;%
{\expandafter\edef\csname #1\ind@x{#2}{#3}\endcsname{#4}}
\newcount\accumulat@r
\def\addto#1[#2,#3]+=#4;%
{\accumulat@r=\value #1[#2,#3]  \advance\accumulat@r by #4
  \assign #1[#2,#3]=\the\accumulat@r;}
```

We do bars as follows: in addition to `\RiCii` etc which define the contents of a square, there is a bar code `\BarRiCii` etc with the following meaning:

0: no bar above or to the left of the square

1: bar above the square

2: bar to the left of the square

3: bars above and to the left of the square

At the start, all bar codes are set to 3; the bars between letters are erased as we go along.

3.2 Macros used while processing the clues

\lettercount
\Barcode

As we move along the letters of the answer, we count the number of letters processed in `\lettercount`. The code for the bar to be erased (1 or 2) is put into `Barcode` before we start on each answer.

```
\newcount\lettercount
\newcount\Barcode
```

\nextletter
\nextlet

To determine how much space is required for the light corresponding to an answer, we need to cycle through each of the characters of the answer individually; this macro is called with two parameters — the first indicates the current setting direction (and thus accesses one of the counters `\Across` or `\Down`), whilst the second consists of the characters forming the answer followed by the string `\@nil`. When it is called, this "second" parameter is *not* enclosed in braces, so only the first token in it is accessed. The macro calls itself recursively to process the remaining characters until the `\@nil` has been met.

```
\def\nextletter#1#2{%
```

If the next token is `\@nil`, we've finished; the `\let` ensures that its parameter will be discarded (through the LaTeX internal command `\@gobble`) and the recursion will then unravel.

```
  \ifx#2\@nil \let\nextlet=\@gobble
```

Otherwise, we have another letter of the ⟨*answer*⟩ in `#2`, so we save the letter and check that it does not conflict with an earlier letter in the same square.

```
  \else \immediate\edef\Before{\value Letter[\Down,\Across]}
      \assign Letter[\Down,\Across]=#2;
      \immediate\edef\After{\value Letter[\Down,\Across]}
      \if \Before ! \else \if \Before \After
      \else \errhelp{You mistyped an answer, or miscounted the coordinates.}
      \errmessage{Letter "\After" conflicts with earlier "\Before"
         in row \the\Down, column \the\Across}
      \fi \fi
% If the letter is not the first letter of its answer, we erase a bar.
     \ifnum \lettercount>0 \addto Bar[\Down,\Across]+=-\Barcode; \fi
% Advance another position in the current direction.
      \@gobble{#2} \advance#1 by \@ne \global\advance \lettercount by \@ne
```

After we've processed this letter, we want to call this routine recursively to process the remaining letters (if any)...

```
    \let\nextlet=\nextletter
  \fi
```

This is where we either exit from the recursion (and `\@gobble` the `#1` parameter) or call the macro recursively to process the next character; the direction has to be passed on as the first parameter for `\nextletter` or `\@gobble`.

```
\nextlet{#1}}
```

3.3 The `\clue` Command

`\clue` Well, here it is at last. We start off by extracting the parts (if any) which form the ⟨*clue_number*⟩ parameter. We will therefore have the purely numeric first portion of the ⟨*clue_number*⟩ in `\cluenumber`.

```
\def\clue#1#2#3#4#5#6#7{%
```

The x and y coordinate counters are set from the ⟨*column*⟩ and ⟨*row*⟩ parameters.

```
  \Across=#3 \Down=#4
```

The clue may be spread over more than one number, but the square takes only the first of these.

```
  \findnumber{#1}
```

Save the clue number, checking that it was not previously defined to be something else.

```
  \immediate\edef\Before{\value Number[\Down,\Across]}
  \assign Number[\Down,\Across]=\cluenumber;
  \immediate\edef\After{\value Number[\Down,\Across]}
  \ifnum \Before>0 \ifnum \After=\Before
  \else \errhelp
     {You probably made a mistake when typing in one of the clues.}
     \errmessage{Number {\After} conflicts with earlier {\Before}
      in row \the\Down, column \the\Across}
     \fi \fi
```

We now examine the second (⟨*Across/Down*⟩) parameter of the `\clue` command to determine whether this is an Across or Down clue. The clue's ⟨*text*⟩ and ⟨*help*⟩ information is then written[4] to the appropriate auxiliary file, and note taken of the direction in which the ⟨*answer*⟩ should be set into the light of the grid.

Firstly we deal with writes to the `.acr` file, if the ⟨*Across/Down*⟩ parameter is the letter '`A`'. In this case, the counter to be incremented is `\Across`.

```
  \ifx#2A
    \global\Barcode=2
    \ifthenelse{\equal{\@empty}{#7}}{}{
      \ifx\tf@acr\relax\else
        \@writefile{acr}{  \item[#1] #6 (#7)}%
      \fi}
    \let\Direction=\Across
  \else
```

[4] The writes only take place if the output files exist, and the ⟨*help*⟩ parameter is non-empty.

If this parameter is the letter 'D', writes go to the `.dwn` file, and the `\Down` counter is incremented.

```
    \ifx#2D
      \global\Barcode=1
    \ifthenelse{\equal{\@empty}{#7}}{}{
      \ifx\tf@dwn\relax\else
        \@writefile{dwn}{  \item[#1] #6 (#7)}%
      \fi}
      \let\Direction=\Down
    \else
```

If this ⟨*Across/Down*⟩ parameter is not one of the two permitted characters, an error message is issued.

```
      \errhelp{The second parameter of the \string\clue\space
               command must be 'A' or 'D'}
      \errmessage{Illegal direction (#1) specification
                  for \string\clue.}
    \fi
  \fi
```

Now we can call `\nextletter` which will cycle through all the letters of the ⟨*answer*⟩ until meeting the token `\@nil`. It will save the current letter and recursively call itself until no letters are left. Finally, we ensure that the newlines after `\clue` commands don't lead to unwanted spaces being typeset.

```
  \global\lettercount=0
  \nextletter{\Direction}#5\@nil
  \ignorespaces
}
```

3.3.1 Finding the clue number to be set in the light

`\findnumber` We mentioned earlier that clues with solutions which occupy more than one light require a special format for specifying their ⟨*clue_number*⟩. If this form is required, the number of the current light is given first, with the remaining text (as it is required to be set) following, separated from the first number by some non-digit character.

The macro `\findnumber` is called with the entire ⟨*clue_number*⟩ parameter passed to `\clue` and sets `\cluenumber` to expand to the first or only number found in that parameter (which should then appear in the first square of the light).

`\clueNumber` Of course, we require a counter in which to attempt to assemble that number:

```
\newcount\clueNumber
```

`\special@gobble` This macro is used by `\findnumber` to discard the unwanted portion (if any) of a ⟨*clue_number*⟩, including the special termination token.

```
\def\special@gobble #1\@nil{}
```

The following mechanism to separate the first (or only) number from the remainder was suggested by Frank Mittelbach of the University of Mainz, and replaced about two pages worth of code.

```
\def\findnumber#1{%
```

We attempt to assign the ⟨*clue_number*⟩ parameter to the `\clueNumber` counter: only that portion consisting purely of digits will actually be assigned. The remainder, if any, including the special terminator sequence `\@nil` is then discarded by the `\special@gobble` command:

```
  \afterassignment \special@gobble \clueNumber=0#1 \@nil
```

If the user did not provide a valid ⟨*clue_number*⟩ (i.e. something starting with a digit), then `clueNumber` will have zero assigned to it. This will cause a diagnostic message at the stage when it is checked that the clue numbers run consecutively in reading order from 1 upwards.

This completes `\findnumber`.

```
  \ifnum\clueNumber=0
    \errhelp{The first parameter of the \string\clue\space command
             must commence with a digit}
    \errmessage{Illegal clue number (#1) specified
                for \string\clue.}
  \fi
}
```

`\cluenumber` This macro merely produces the number as saved in `\clueNumber`.

```
\def\cluenumber{\the\clueNumber}
```

3.4 Populating the Crossword Grid

`\blackenrow` For each column in a row we create a black square by setting the letter in the square to `!`. We also initialize the `Number` and `Bar` macros.

Before starting the inner loop, we need to save the definition of `\body` which was created for the outer loop. We cannot do this by creating a new block, since that would require that each square be defined globally, which might give rise to save stack overflow problems.

Actually with LaTeX 2ε we have to save `\iterate` as the internals of `\loop` have been streamlined. With the old code only the first row was updated—this is the price for using an unpublished implementation feature.

```
\def\blackenrow{\let\saveditерate\iterate
  \loop\relax\ifnum\Across>\z@
    \assign Bar[\Down,\Across]=3;
    \assign Letter[\Down,\Across]=!;
    \assign Number[\Down,\Across]=0;
```

We then shift ourselves back to the next column to the left and iterate. If we've reached the end of this inner loop, we re-establish the definition of `\body`.

Again with 2e this has to be `\iterate`.

```
    \advance\Across by \m@ne
```

```
  \repeat
  \let\iterate\savediterate
}
```

`\SetUpGrid` This macro creates an empty grid of the appropriate size.

```
\def\SetUpGrid#1#2{%
```

We firstly make a note of the ⟨*gridcols*⟩ parameter in the `\gr@dcols` counter, from which the width and height of the grid may be computed. We also set the `\p@csize` counter to be one greater than `\gridc@ls` to save our recomputing this quantity many times over. Ditto mutatis mutandis for ⟨*gridrows*⟩.

```
  \gridr@ws=#1\gridc@ls=#2
  \p@rsize=#1 \advance\p@rsize by \@ne
  \p@csize=#2 \advance\p@csize by \@ne
  \typeout{Grid has #1 rows and #2 columns.}
```

Right, this is where we start to generate the grid itself. We start at the bottom edge, because TEX loops are easiest if counting down to zero. Therefore, the `\Down` counter is set equal to the highest row number attainable.

```
  \Down=\gridr@ws
```

We now start a loop, so the following code will be repeated for each row of the grid in turn. As with the rows, we process the columns from highest address to lowest, so the `\Across` counter is also set to the highest column attainable.

```
  \loop
    \Across=\gridc@ls
```

Provided we haven't decremented down to the 0th row, we start off the inner loop to process each column: this is done by invoking a separate macro — the alternative to which would be to enclose the inner loop in a group, which would require the use of global definitions for each square.

```
    \ifnum\Down>\z@
        \blackenrow
```

Afterwards we move ourselves up one row, and iterate for the next row.

```
        \advance\Down by \m@ne
  \repeat
}
```

And that's the end of `\SetUpGrid`!

`\TestAnswers` This macro interacts with the user, if necessary, to get a yes or no indication of whether the answers shall be written into the grid. No check is made that the user has entered a valid response, but the use of `\answer` is such that any answer apart from a '`y`' (in upper- or lower-case) is treated as if it were '`n`'.

`\f@rst` To determine what parameter has been provided, or the response elicited, we will require a little macro to pass on the first token of a list terminated by a full stop.

```
\def\f@rst#1#2.{#1}
```

We commence by lower-casing the given parameter, setting the lower-cased version into the macro `\answer`.

```
\def\TestAnswers#1{\edef\next{\def\noexpand\answer{#1}}%
  \lowercase\expandafter{\next}%
```

We can then extract just the first character

```
  \edef\answer{\expandafter \f@rst \answer n.}%
```

The we determine whether it's the letter '`y`' or '`n`'...

```
  \if\answer y \else \if\answer n \else
```

If the ⟨*visible*⟩ parameter isn't either of these, we ask the user to give us an answer!

```
    \typein[\answer]{Make answers visible? [Y/N]: }\fi
  \fi
```

OK, `\answer` now contains some response; let's upper-case it and extract just its first character

```
  \edef\next{\def\noexpand\answer{\answer}}%
  \uppercase\expandafter{\next}%
  \edef\answer{\expandafter \f@rst \answer n.}%
}
```

3.5 Setting the Grid

`\putletter` These macros put letters, numbers and bars at the current `[\Down,\Across]` position. Note that `\putletter` would be redefined to `\relax` if no actual letters are required.

```
\def\putletter{{\put(\Across,-\Down){\makebox(1,1)
   {\sffamily \value Letter[\Down,\Across]}}}}
```

`\putnumber` The `\putnumber` macro checks whether the clue number is 0 (which means an unnumbered square) and whether the clue numbers are in sequence. To this end, it makes use of global `\oldnumber` and `seqtest`.

```
\newcount\oldnumber \oldnumber=0
\newif\ifseqtest \seqtestfalse
\def\putnumber{\immediate\def\Number{\value Number[\Down,\Across]}
   \ifnumberit \ifnum\Number>0
   \put(\Across,-\Down){%
```

To insert the clue number, we generate it within a sub-picture, of size equal to one square.

```
    \begin{picture}(1,1)(0,0)
```

We stick the number in the top-left corner of an (invisible) box which fills the central 81% of the area. In the process, we check whether the clues are properly numbered in reading order.

```
      \put(0.05,0.05){\makebox(0.9,0.9)[tl]{\numbersize\Number}}
    \end{picture}}
  \ifnum \oldnumber>0 \advance\oldnumber by\m@ne
    \ifnum \oldnumber=\Number {\seqtesttrue} \else
```

```
      \ifseqtest
      \errhelp{Clues should be numbered consecutively from top left
               to bottom right.}
      \errmessage{Found clue number \Number, but expected \the\oldnumber}
```

If a clue number was mistyped, it will generate two out-of-sequence errors. We suppress the second of these.

```
      \seqtestfalse
      \else\seqtesttrue\fi
    \fi
  \fi
  \global \oldnumber=\Number
\fi \fi
}
```

`\putbars` If there are two adjacent light squares not belonging to the same light, we draw a bar between them. We use two counters for that: one for the previous square, so we can test for its blackness; one for the changing value of the barcode for the current square.

```
\newcount \pr@vious
\newcount \c@de
\def\putbars{ \linethickness{\barwidth\unitlength}
  \c@de=\value Bar[\Down,\Across]
  \ifnum \c@de>1 \advance\c@de by -2
     \ifnum \Across>1
       \pr@vious=\Across \advance \pr@vious by -1
       \immediate\def\pr@vlet{\value Letter[\Down,\pr@vious]}
       \if \pr@vlet ! \else \put(\Across,-\Down){\line(0,1){1}}
   \fi \fi \fi
  \ifnum \c@de>0
     \ifnum \Down>1
       \pr@vious=\Down \advance \pr@vious by -1
       \immediate\def\pr@vlet{\value Letter[\pr@vious,\Across]}
       \if \pr@vlet ! \else \put(\Across,-\pr@vious){\line(1,0){1}}
   \fi \fi \fi
}
```

`\FinishGrid` Now we can process all the stored macros which define the appearance of each square in the grid, and thus generate the printed version thereof, using LaTeX's picture environment.

This command makes appropriate redefinitions of some macros which produced different effects during the filling of the grid.

```
\def\FinishGrid{%
```

If the customer doesn't want the letters put into the grid, then we need only let `\putletter` do nothing. Since `\FinishGrid` is invoked inside the environment, the original definition will be restored on exit.

```
  \if\answer Y \else \let\putletter\relax \fi
```

Now we come to the actual body of `\FinishGrid`.
We start off at the bottom-most row of the grid...

```
\Down=\gridr@ws
```

The whole grid is created in a centered `\hbox` in a picture environment. By offsetting the origin negatively, we can address each row by simply negating the y coordinate; thus column x in the highest row is $(x, -1)$.

```
\centerline{%
  \begin{picture}(\p@csize,\p@rsize)(1,-\p@rsize)
```

We now cycle through each of the rows. The first thing we output is a horizontal rule of the full width of the grid, one such rule being generated for each row of the grid, providing the horizontal lines across vertical lights.

```
  \loop\ifnum\Down>\z@
    \put(1,-\Down){\line(1,0){\the\gridc@ls}}
```

Now we are about to cycle across all the columns of the current row; again, it's convenient for us to work backwards to the left...

```
    \Across=\gridc@ls
```

To do this we need an inner loop; this has to be inside a group so as to isolate the effects of its `\repeat` command.

```
    {\loop \ifnum\Across>\z@
```

A black square is represented by `!` instead of a letter. We need an `\immediate` definition to force the expansion so we can test using `\if`. This test cannot be made inside `\putletter` because that macro may have been redefined when the lights are empty.

```
      \immediate\def\lett@r{\value Letter[\Down,\Across]}
      \if \lett@r ! \put(\Across,-\Down){\rule{\unitlength}{\unitlength}}
      \else \putletter \putnumber \putbars
      \fi
```

Now we advance to the next column and iterate. That's the end of the inner loop for each of the columns of the current row.

```
       \advance\Across by \m@ne
     \repeat
    }%
```

Now we can decrement down to the next row and iterate through the rows. In the process, we encounter all the clue numbers in opposite order, therefore the last clue number found should be 1.

```
    \advance\Down by \m@ne
  \repeat
  \ifthenelse{\equal{\the\oldnumber}1}{}
   {\ifnumberit\errhelp{Clues should be numbered consecutively
      from top left to bottom right.}
    \errmessage{First clue number is \the\oldnumber, not 1} \fi}
```

We've so far drawn a horizontal line *under* each of the rows; the next `\put` draws a final line *above* the top-most row.

```
    \put(1,0){\line(1,0){\the\gridc@ls}}
```

Similarly, a short loop can draw vertical rules at the left-hand edge of each of the columns, starting with a line on the left of an imaginary column to the right of the whole grid, which will therefore form a line to the right of the final column.

```
    \Across=\p@csize
    \loop\ifnum\Across>\z@
      \put(\Across,0){\line(0,-1){\the\gridr@ws}}
      \advance\Across by \m@ne
    \repeat
```

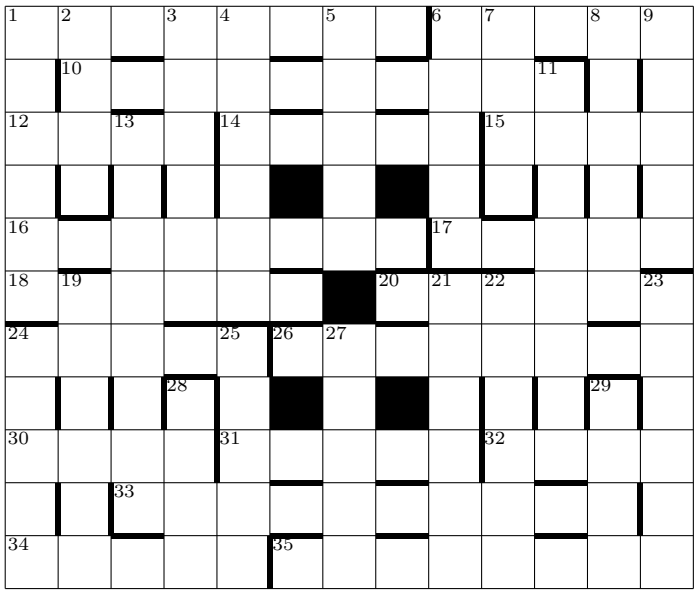
And that completes the picture.

```
    \end{picture}%
  }%
}
```

⟨/package⟩

4 A hard crossword with blocks and bars

T-shirt design by Dirk Laurie



5 27, and that goes for 17 11 25 and 24A 10 too, I'm sure, so the unclued lights should be entered in red. The other lights are all in the rather ancient edition of Chambers consulted by the composer.

ACROSS

1 Murderer's lip in an evil act (8)
6 Places where vehicles without resistance get bogged down? (5)
12 Perhaps 27 die here (4)
14 Obstinate at first and after a smack (5)
15 Seduced girl, as rumoured (4)
16 Paid an unspecified amount in Johannesburg (8)
17 & 11D & 25D (See preamble)
18 Butt gives information in a peeve (6,hyphenated)
20 Big folk appearing in spots last month (6)
24 & 10 (See preamble)
26 Wince after European has knocked back medicine, leaving out some (8)
30 Recidivist is back, getting old in here (4)
31 Unwilling to receive a small copper pot (5)
32 Primal wilderness beside a person from the East "is Paradise —" (4)
33 Denizen of Africa to suffer in the heat when dose insects get in (10)
34 Turn around at home and peek outside, like 18? (5)
35 Such as these two from the back going ahead in copious lovemaking (8)

DOWN

1 A disc full of French horn music (6)
2 Drams served up in Highland drizzle (4)
3 A gadfly with well-separated feet etc (6,two words)
4 Not many shooting up, in fact only one (6)
5 & 27 (See preamble,three words)
6 Tree that will, having had root extracted at first, turn into mineral! (5)
7 Give me a response that I can hear: it's no longer a crisis (4)
8 What unreformed head wears is right (6)
9 Dirk's parrot, found in polar regions (5)
13 Saying "Be with you shortly", lassie goes inside house (8)
19 Invest ten rand judiciously (6)
21 Coin Dirk lost on the way up; Henry gets involved (6)
22 Peek! (6)
23 Neutral sounds and trite sayings arise all around Switzerland (6)
24 Servants genteel at heart? One of them is the butler at Blandings Castle! (5)
28 Serving up only half a beer is a mistake (4)
29 Footplay before strong no-trump not allowed (4)

5 Sample input files

Finally, here's the input which produced the crossword in Figure 1

```
\begin{crossword}{15}{N}
   \input{grid1}
\end{crossword}
```

```
⟨*grid0⟩
\clue{1}{A}{3}{1}{PERAMBULATOR}{Stroller carrying the baby}{12}
\clue{8}{A}{1}{3}{IMPASSE}{Eastern note returned about predicament resulting
                           in deadlock}{7}
\clue{9}{A}{9}{3}{TERMINI}{Many ends brought back in one minute soak}{7}
\clue{11}{A}{1}{5}{TIEPOLO}{Venetian painter to draw Spanish gipsy dance?}{7}
\clue{12}{A}{9}{5}{ARAMAIC}{Semitic language of one in charge of following a
                            sheep}{7}
\clue{13}{A}{1}{7}{ENSUE}{Follow three directions and bend eastward}{5}
\clue{14}{A}{7}{7}{IMPLETION}{Filling feeble-minded person who lost his head
                              about one}{9}
\clue{16}{A}{1}{9}{SURPRISED}{Rip apart with duress taken unawares}{9}
\clue{19}{A}{11}{9}{DUNCE}{Stupid person from hill church}{5}
\clue{21}{A}{1}{11}{ERRATUM}{Published correction --- return walled-up
                             sailor}{7}
\clue{23}{A}{9}{11}{LIASSIC}{Lower Jurassic sea trip backwards [thus]}{7}
\clue{24}{A}{1}{13}{SABBATH}{Midnight meeting of witches at hospital on rest
                             day?}{7}
\clue{25}{A}{9}{13}{ISOTRON}{Sort ion with an accelerator?}{7}
\clue{26}{A}{2}{15}{STANDINGROOM}{Substitute husband relegated here at
                                  crowded wedding}{8-4}
\clue{1}{D}{3}{1}{PEPPERS}{Showers band with lonely hearts?}{7}
\clue{2}{D}{5}{1}{RESTORE}{Concerning money saved --- reinstate to former
                           owner}{7}
\clue{3}{D}{7}{1}{MNEMONICS}{No-one in higher degree produces memory
                             joggers}{9}
\clue{4}{D}{9}{1}{ULTRA}{Enigmatic secret of WW\,I\kern-.1em I?}{5}
\clue{5}{D}{11}{1}{ACREAGE}{Current about time flows over measured area}{7}
\clue{6}{D}{13}{1}{ORIGAMI}{Maybe magic endlessly raised before one with
                            Japanese artistic skill}{7}
\clue{7}{D}{1}{2}{LISTLESSNESS}{Enroll fewer once before head suffering a
                                languid malaise}{12}
\clue{10}{D}{15}{3}{INCANDESCENT}{White-hot Peruvian reorganized objects
                                  covering small coin}{12}
\clue{15}{D}{9}{7}{PEDALLING}{Penny led a rearranged fish engaging in cyclic
                              activity!}{9}
\clue{17}{D}{3}{9}{RAREBIT}{Welsh toast? An unusual drill!}{7}
\clue{18}{D}{5}{9}{RETRAIN}{Keep in mind, take in, and acquire new
                            skills}{7}
\clue{19}{D}{11}{9}{DIABOLO}{Twice a high ball thrown up round a two-headed
                             top}{7}
```

```
\clue{20}{D}{13}{9}{NOSTRUM}{No good man with spirit offers quack remedy}{7}
\clue{22}{D}{7}{11}{MAHDI}{I would shortly overact, sent up, and the
                           downfall of Gordon}{5}
```
⟨/grid0⟩

⟨*grid1⟩
```
\clue{8}{A}{1}{2}{SWAM}{Points a thousand tested for witchcraft.}{4}
\clue{9}{A}{6}{2}{OHO}{Gourmet's triumphant cry on finding middle-cut
                       Pacific salmon!}{3}
\clue{10}{A}{10}{2}{ICECAP}{One hundred stride backwards across a
                            Pole.}{3-3}
\clue{11}{A}{1}{4}{BOPEEP}{Fifties' jazz record about Eastern childs'
                           play.}{2-4}
\clue{12}{A}{8}{4}{SCHEDULE}{Timetable created by editor in synagogue
                             of Spain.}{8}
\clue{13}{A}{1}{6}{THALASSOGRAPHER}{The dialect a girl mixed up tangle
                                    around symbolic diagram used by
                                    maritime student!}{15}
\clue{15}{A}{1}{8}{HAIRPIN}{Wire fastening bent road narrowly.}{7}
\clue{17}{A}{9}{8}{UMBRAGE}{Hammerhead consumes German company
                            and casts a shade.}{7}
\clue{20}{A}{1}{10}{SCALENETRIANGLE}{Strange cel alien chops to make a
                                     figure with odd sides.}{7,8}
\clue{23}{A}{1}{12}{AMOUNTED}{Sounds like a hoarse editor came to in
                              total!}{8}
\clue{25}{A}{10}{12}{ALLEGE}{Assert without proof everyone, for
                             example, English.}{6}
\clue{26}{A}{1}{14}{FLORAL}{Flourished examination of flowers.}{6}
\clue{27}{A}{8}{14}{NIL}{Floor covering discards fuming sulphuric acid
                         and returns to nothing.}{3}
\clue{28}{A}{12}{14}{TACE}{``Latin for a candle'' to be silent note
                           about aircraftman?}{4}
\clue{1}{D}{2}{1}{SWOOSH}{Water rush noise disturbs show so!}{6}
\clue{2}{D}{4}{1}{IMPELLER}{Mischievous child with cloth measure
                            hesitates to assemble rotor.}{8}
\clue{3}{D}{6}{1}{COMPASSIONATELY}{Mercifully inclined to pass round
                                   ten? Nay, about short blower!}{15}
\clue{4}{D}{8}{1}{TORSION}{Sort ion? An isotron gives it a new
                           twist!}{7}
\clue{5}{D}{10}{1}{DITHYRAMBICALLY}{Wildly and boisterously rearrange
                                    Billy May Third, roughly.}{15}
\clue{6}{D}{12}{1}{SENDUP}{Satirical book or film---give odds about
                           revision of ``Dune''?}{4-2}
\clue{7 {\noexpand\rm\&} 24}{D}{14}{1}{PALL}{Premier took in a Lord
                                    Lieutenant and all played an
                                    old game in London street.}{4-4}
\clue{14}{D}{14}{6}{ERG}{Work expended in power games?}{3}
\clue{16}{D}{2}{8}{ARC}{A church circle (or part of one).}{3}
\clue{18}{D}{12}{8}{RINGLETS}{Encircle hindrances under hair in long
                              curls.}{8}
```

```
\clue{19}{D}{8}{9}{STUDENT}{Boss over oto\-rhino\-laryng\-o\-logy
                            department undergraduate.}{7}
\clue{21}{D}{4}{10}{LOUVRE}{Old dovecote in Parisian museum.}{6}
\clue{22}{D}{14}{10}{LEGACY}{Like ornamental fabric, for example,
                             that's in bequest.}{6}
\clue{24}{D}{2}{12}{MALL}{}{See {\bf 7}}
```
⟨/grid1⟩

⟨*exa0⟩
```
\documentclass{article}
\usepackage{crosswrd}
\pagestyle{empty}
\setlength\topmargin{-1in}
\setlength\textheight{10.7in}

\begin{document}
    \begin{crossword}{15}{?}
    \input{grid0}
    \end{crossword}
\end{document}
```
⟨/exa0⟩

⟨*exa1⟩
```
\documentclass{article}
\usepackage{crosswrd}
\pagestyle{empty}
\setlength\topmargin{-1in}
\setlength\textheight{10.7in}
\begin{document}
    \section*{A Sample Crossword}
    \textit{(Chamber's Twentieth Century dictionary)}
    \begin{crossword}{15}{N}
      \input{grid1}
    \end{crossword}
    \begin{crossword*}{15}{Solution to grid 0}
      \input{grid0}
    \end{crossword*}
\end{document}
```
⟨/exa1⟩