User's Guide to the LaTeX Calendar Bundle*

Frank G. Bennett, Jr.

February 18, 2004

Contents

*This file is version number 3.1. It was last revised on 1998/01/17 17:11:27.

Preface

Welcome to the LaTeX Calendar bundle. This bundle will produce calendars in a variety of formats, in any language supported by LaTeX 2_ε, simply, quickly, and with enough flexibility to make the little "personal calendars" that ship with Windows 95 and the like beg for mercy. It just might be the only calendaring system you need.

This manual explains how to use it.

1 System Preparation

The first thing you must do is be sure that you have all of the necessary tools to hand. Before using the styles in the LaTeX calendar package, you should first check that your LaTeX installation includes all of the utilities in the `graphics` subdirectory under `packages` in the main LaTeX 2_ε distribution directory on CTAN, and the `longtable` and the `multicol` packages from the `tools` subdirectory of `packages`. You will also need everything in the `calendar` package itself. With these items in hand, the package should function smoothly in accordance with the documentation given here and in the individual `*.dtx` files. If you have problems, please contact me on `bennett@nomolog.nagoya-u.ac.jp`.

2 Creating Calendars

There are nine calendar styles in the bundle. The simplest possible document for each is shown in Figure 1. It is worth spending a little time studying these examples to see what the essential elements are, and what can vary from one style to another. Some are packages, invoked through the `\usepackage` tag. Others are classes, invoked via the `\documentclass` tag at the top of the document. In either case, the calendar itself takes the form of a LaTeX environment. The name of this environment sometimes differs from the name given to `\usepackage` or `\documentclass`, because the operating system on which the largest software company in the world was built cannot cope with more than eight characters in a file name.

There are *always* two extra arguments after the `\begin` tag that opens the environment. The first extra argument contains formatting instructions and other options that modify the behavior of the style. We will refer to these as *calendar options*. A few calendar options are common to all calendar styles, while others are specific to a particular style. In the examples, the first extra argument is always empty because (get ready for this one) calendar options are optional. A full explanation of all calendar options for all styles is given in section 4 below.

The second extra argument gives names of data files containing lists of events or appointments, together with the names of any colors to be associated with the entries in each data file. In the examples, this second argument is blank because a calendar can be created without including any events or appointments. See section 3 below for a discussion of how to manage data lists and include them in calendars.

The only other component of a complete calendar is a *controlling date* or a *controlling range* of dates. As you can see from the examples, this is the only

```
\documentclass{article}
\usepackage{timesht}
\begin{document}
\begin{timesheet}
{}
{}
 Jan 1 1997
\end{timesheet}
```

```
\documentclass{weekly}

\begin{document}
\begin{weekly}
{}
{}
 Jan 1 1997
\end{weekly}
```

```
\documentclass{monthly}

\begin{document}
\begin{monthly}
{}
{}
 Jan 1 1997
\end{monthly}
```

```
\documentclass{article}
\usepackage{yearly}
\begin{document}
\begin{yearly}
{}
{}
 Jan 1 1997
\end{yearly}
```

```
\documentclass{article}
\usepackage{evntlist}
\begin{document}
\begin{eventlist}
{}
{}
 Jan 1 1997
\end{eventlist}
```

```
\documentclass{timetabl}

\begin{document}
\begin{timetable}
{}
{}
 Jan 1 1997
\end{timetable}
```

```
\documentclass{article}
\usepackage{hmonth}
\begin{document}
\begin{htmlmonth}
{}
{}
 Jan 1 1997
\end{htmlmonth}
```

```
\documentclass{article}
\usepackage{hlist}
\begin{document}
\begin{htmllist}
{}
{}
 Jan 1 1997
\end{htmllist}
```

```
\documentclass{article}
\usepackage{hml}
\begin{document}
\begin{htmlmonthlist}
{}
{}
 Jan 1 1997
\end{htmlmonthlist}
```

Figure 1: Minimal documents for each calendar style

thing written inside the environment, and it is written in a human-readable form. The bundle is pretty forgiving in the way it reads dates; the elements of a simple date can be written in any order (that is to say, `Aug 1 1997`, `1 Aug 1997` and `1997 Aug 1` will all work equally well). Only the first three letters of the month are read, but any extra characters are quietly ignored (so you can say `August 1 1997` instead of `Aug 1 1997`). And the first character of the month can be either upper-case or lower-case (so `1 aug 1997` is okay too).

The finished calendar will be composed so that it includes the controlling date. This may be expressed as a *range* rather than, as in the examples, a *pinpoint date*. A range is simply two pinpoint dates separated by the word `to`. If a range is used (for example, `jan 1 1997 to 1 august 1997`), the finished calendar will be composed so that it includes both the starting and the ending date of the controlling range.

We have now completed an overview of how calendars are written. All that remains is to examine (a) how data files are written and included in documents (see section 3 below), and (b) what options are available to modify the behavior of each of the styles (see section 4 below).

3 Data Files

Most commercial calendars come with pre-printed notes of the major holidays and other significant events. Calendars created with the LaTeX Calendar Bundle can contain pre-printed notes of any events or appontments that you like. Events or appointments are incorporated into a calendar from separate files that have the extension `.cld`. The names of the files, less the extension, should be listed, separated by commas and without spaces, in the second argument to the calendar environment, as show in Figure 2. In the next section, we will examine the content of the data files themselves.

```
\documentclass{weekly}
\begin{document}
\begin{weekly}{}{firstlist,secondlist}
  Jan 1 1997
\end{weekly}
\end{document}
```

Figure 2: Data file names

3.1 Pinpoint Date Syntax

The simplest type of `.cld` file is a list of dates, each followed by a note in curly braces. Such a file is shown in Figure 3. The month is specified as a word (the first three letters of which will be recognized), the day as an integer between `1` and `31`, and the year as a four-digit integer. The items may come in any order, and the first letter of the word may be in upper or lower case.

The text given in curly braces should be a brief note describing the event or appointment. This can be followed by a longer note in square braces. Styles will use this longer text where appropriate and physically possible. An example is given in Figure 4.

```
 Jan 1 1997 {New Year's Day}
 Jan 1 1997 {The Day After New Year's Day}
```

Figure 3: Simple pinpoint entries

```
jan 1 1997 10:00-11:30 {Smith}
  [Smith: \textit{Reflections on Principle in the work
   of Ambrose Bierce}]
```

Figure 4: Verbose description in square braces

Finally, a time or a range of times may be added to pinpoint entries. In this manual, I refer to entries that do not specify a time as "events", and to those that do specify a time as "appointments", but there is no other difference between the two. They may be intermixed within a single `.cld` file, and in any order. There should be no space in the time string. See Figure 5 for an example of what appointment entries look like.

```
jan 1 1997 8:00 {Happy New Year}
 [Take stock of New Year hangover and damage to home
  and hearth]
jan 1 1997 10:00-16:00 {New Year cleaning}
```

Figure 5: Appointment entries

3.2 Recursive Date Syntax

Some events happen repeatedly. Birthdays happen every year. Classes happen every week during school terms except on holidays. Phases of the moon recur. It is not very efficient to use repeated pinpoint entries for such events, so the LaTeX Calendar Bundle allows you to write such dates descriptively. This manual refers to dates defined in this way as *recursive* dates.

```
% file: class.cld
% The schedule for my Spanish class
range curyear {Current year}
every thurs {Spanish class}
```

Figure 6: `class.cld`

```
% file: curyear.cld
% A range including all dates in the current year
January 1 1997 to December 31 1997 {The year}
```

Figure 7: `curyear.cld`

Figures 6 and 7 show the content of two `.cld` files, which together define a

simple recursive date. The file `class.cld`, shown in Figure 6, will produce a note saying "Spanish class" for every Thursday within the range given in the file `curyear.cld`. It is worth pausing for a few moments to study how these two files are constructed.

Note that the `range` declaration takes a filename (less the `.cld` extension), followed by a comment in curly braces. The comment will not be printed in the calendar output, so the braces can be left empty if desired. The file called by the `range` declaration should contain one or more ranges, each of which is composed of a starting and an ending date separated by the word `to`. Each range should be ended with a comment (possibly, as in the example, an empty comment) in matched curly braces.[1]

The recursive date itself always begins with the word `every`, followed by a description of conditions to be repeated throughout the range. Figure 8 gives examples of all of the conditions that can be specified. The only entry that is not self-explanatory, I think, is the `every third` (or `first`, or `second`) `Tuesday` example. This puts an entry on the `n`*th* day of the specified week of the month throughout the specified range. Note also that times can be specified for all recursive entries (that is to say, you can specify recursive appointments as well as recursive events).

```
every day          {Rejoice if you can}
every day 8:00     {Wake up}
every 20           {Payday}
every 20  12:00    {Buy big lunch}
every 1 April      {All Fools Day}
every 1 april 8:00 {Practical joke time!}
every third tuesday          {Village council}
every third monday 2:00-3:00 {That appointment}
every Tuesday 3:00-4:00 {Aerobics}
every Tuesday           {Happy Tuesday!}
every other week from 12 dec 1997       {Kite flying}
every other week from 12 dec 1997 17:00 {Repair kite}
```

Figure 8: Examples of all possible recursive entry types

There is one common situation with which the simple recursive date illustrated in Figures 6 and 7 cannot cope. Suppose I want to define a recursive date that will put an entry in my diary for classes that I teach on a regular basis. I might try to do this using an entry like that in Figure 9.

The problem is that the simple ranges recorded in `schoolterms.cld` will not account for holidays, and I do not plan to teach classes during holidays. To cope with this little difficulty, the LaTeX Calendar Bundle provides the `holiday` declaration. As you can see in Figure 10, the syntax of this declaration is the same as that of the `range` declaration. The separate file `ukhols.cld` should contain a list of pinpoint dates which are to be excluded when recursive dates are calculated.

[1] The starting date may also be followed by a comment in curly braces. It is often useful to associate a comment with the starting and ending dates of a range. See Figure 11 and the discussion at the end of this section.

```
range schoolterms {Terms}
every Tuesday 10:00-12:00 {Jse Lec}
  [Japanese Law lecture]
every Tuesday 13:00-14:00 {Prop Tut}
  [Property Law tutorial]
```

Figure 9: First attempt to record my teaching dates

```
range schoolterms {Terms}
holiday ukhols    {United Kingdom holidays}
every Tuesday 10:00-12:00 {Jse Lec}
  [Japanese Law lecture]
every Tuesday 13:00-14:00 {Prop Tut}
  [Property Law tutorial]
```

Figure 10: A better attempt to record my teaching dates

The text enclosed in curly braces that follows a `range` or `holiday` declaration (see Figures 6, 9 and 10), and the comments in curly braces inside the `.cld` files called by these declarations (see Figure 7) will not appear in the text of printed calendars. Sometimes it is *useful* to print a note of the beginning and ending dates of a range, or the dates of holidays. In this case, the file containing the relevant range (which should have notes associated with the starting an ending dates) should be named in the second argument to the calendar environment, as a normal input file. Figure 11 gives an example of a fully-commented range file.

```
% file: ug-current.cld
17 sep 1997     {First SOAS term begins}
 to 12 dec 1997 {First SOAS term ends}
5 jan 1998      {Second SOAS term begins}
 to 20 mar 1998 {Second SOAS term ends}
20 apr 1998     {Third SOAS term begins}
to 5 jun 1998   {Third SOAS term ends}
```

Figure 11: Ranges with comments

3.3 Color Support

You can make calendars that are more informative and easier to read by printing entries of different types in different colors. To do this, you must first include color support by calling the `color` package. If the color package has not been configured for your system, you may need to tell it about your driver. If you use `dvips`, it is handy to use named colors and the pallette of colors provided by the `dvipsnames` option. See the documentation to the `graphics` package for further details.

Once color support is in place (and you have a color printer to hand), you can associate a color with the entries in a data file by putting a colon after

```
\documentclass{weekly}
\usepackage[dvips,usenames,dvipsnames]{color}
\begin{document}
```

Figure 12: A header that invokes color support

the name of the file, followed by the name of the color in which you want its entries to appear. See the documentation to the `graphics` package for further information on what color names are available.

```
\documentclass{weekly}
\begin{document}
\begin{weekly}{}{firstlist:red,secondlist:blue}
  Jan 1 1997
\end{weekly}
\end{document}
```

Figure 13: Data file names

3.4 Freezing Data

A price must be paid for the very friendly syntax of the data files used by the LaTeX Calendar bundle; when large amounts of data are involved, it can take a significant amount of time to process a calendar. The `freeze` declaration provides a means of speeding things up. To freeze the data in a file, put the declaration `freeze` (without any following curly braces) immediately after any range or holiday specifications. All subsequent dates generated by the file will be captured in a pre-processed form to a file with the same name, but with the extension `.eve`. The next time a style from the LaTeX Calendar bundle reads that file, the pre-processed `.eve` file will be read instead of the `.cld` file.

```
range year {Current year}
freeze
every second thursday {Working group}
```

Figure 14: An example of the `freeze` declaration

3.5 Composite Lists

Large or complex data sets can be easier to manage if the data is broken down into small chunks. The LaTeX Calendar bundle facilitates this through the `list` declaration, which allows one `.cld` file to incorporate another by reference. The file called by `list` may include either pinpoint or recursive dates. This can be used to create a file representing an entire category of entries. Figure 15 shows the content of a file that I used to represent all of the undergraduate lectures in a university law department in London.

A nice feature of the `list` declaration is that the color associated with the "top" file is inherited by files called from it using `list`. When dealing with

```
list ug-cinl-lecture {}
list ug-cinl-lecture {}
list ug-conflicts-lecture {}
list ug-criminal-lecture {}
list ug-csel-lecture {}
list ug-eml-lecture {}
list ug-equity-lecture {}
list ug-essay-lecture {}
list ug-family-lecture {}
list ug-hr-lecture {}
list ug-iel-lecture {}
list ug-islam-lecture {}
list ug-jsel-lecture {}
list ug-ld-lecture {}
list ug-ld-lecture {}
list ug-lsaa-lecture {}
list ug-lssa-lecture {}
list ug-obs1-lecture {}
list ug-obs2-lecture {}
list ug-pil-lecture {}
list ug-property-lecture {}
list ug-public-lecture {}
```

Figure 15: `ug-lectures.cld`

large or complex data sets, this is a Really Nice Feature. Note, however, that the `freeze` declaration will *not* export any data contained in files referenced using `list`. The `freeze` declaration should therefore be used *only* in files that do *not* contain any `list` declarations.

If your operating system can cope with long filenames, see the `tar` archive file `bigdemo.tgz` in the `calendar` subdirectory on CTAN for a complete working data set that relies on the `list` declaration to organize things in this way.

3.6 Language Support

The LaTeX Calendar bundle can be quickly configured to support any language that LaTeX is capable of handling. By default, calendars are produced using English names of the month and names of days of the week. It can also support *multiple* languages in each of the `*.cld` files on your system, and produce the same calendar with month and day names *and* the entry texts in alternative languages.

The language configuration of the LaTeX Calendar bundle is contained in the file `dates.cfg`. The sample file that ships with the bundle has the following contents:

```
\DeclareCalendarLanguage{English}
 {{Sunday}{Monday}{Tuesday}{Wednesday}{Thursday}{Friday}{Saturday}}
 {{Sun}{Mon}{Tue}{Wed}{Thu}{Fri}{Sat}}
 {{January}{February}{March}{April}{May}{June}{July}{August}{September}
  {October}{November}{December}}
```

```
 {{Jan}{Feb}{Mar}{Apr}{May}{Jun}{Jul}{Aug}{Sep}{Oct}{Nov}{Dec}}

\DeclareCalendarLanguage{German}
  {{Sonntag}{Montag}{Dienstag}{Mittwoch}{Donnerstag}{Freitag}{Samstag}}
  {{So}{Mo}{Di}{Mi}{Do}{Fr}{Sa}}
  {{Januar}{Februar}{M\"arz}{April}{Mai}{Juni}{Juli}{August}{September}%
    {Oktober}{November}{Dezember}}
  {{Jan}{Feb}{M\"arz}{Apr}{Mai}{Juni}{Juli}{Aug}{Sep}{Okt}{Nov}{Dez}}
```

These entries serve two purposes. First, they define the texts that should be use for long and short names of the days of the week and the months for each of the language options that they declare. Second, the *order* of options determines the order of alternative language entries in your `*.cld` files. In the example above, `English` is the default language, but `German` can be specified as an option in your documents, by giving that as a class or package option to the calendar style you are using. You can declare as many languages in your `dates.cfg` file as you like.

Once you have edited the `dates.cfg` file to your satisfaction, and put the languages into the correct order for your needs, you are ready to add multilingual `*.cld` to your files. The first thing to note is that alternative languages are always *optional*. If we are using the sample `dates.cfg` file above, and the `*.cld` file contains only default entries in English, the English entries will be supplied. Nothing will break as a result of specifying an alternative language that has been declared; they are perfectly safe.

Again following the example configuration file above, a German entry can be added to a file by using angle braces instead of curly braces for its short text. Square braces (for a long text) work exactly as they do after the default, curly-brace text:

```
jan 1 1998 {New Year}
           <Neue Yahr>

jan 1 1998 21:00 {New Year's Party} [New Year's Party at Fred's house.]
                 <Neue Yahr>        []
```

3.7 Rotating Text

Two of the style packages (`weekly` and `timetabl`) need to be able to twist text sideways before it is printed. Internally, they use the special facilities of the `graphics` bundle for this purpose. You may want to turn other calendars (such as the `monthly` calendar, for example) sideways in order to make better use of the space on a page. To get full use out of this package, it is therefore important that you have the the `graphics` bundle installed on your system, and that your print driver be one that is capable of rotating text. I don't have access to most of the devices supported by the `graphics` package, but from the code in the package it looks as though only the following drivers support rotation: `dvips`, `dvipsone`, `dvitops`, `pctex32`, `pctexps`, `pubps`, `textures`.

Note that the `graphics` bundle must also be *configured* properly for your system. In essence, this means that the files in the bundle must be told what driver you are using, via a file called `graphics.cfg`. If you have problems, please see the documentation in the file `grfguide.tex`, shipped with the `graphics` bundle.

To rotate an entire calendar (such as `monthly`), use the `lscape` package, and enclose the calendar tags in `landscape` tags:

```
\documentclass{article}
\usepackage{monthly}
\usepackage{lscape}
\begin{landscape}
\begin{monthly}{}{}
  january 1 1998
\end{monthly}
\end{landscape}
```

3.8 Arbitrary Formulæ

Finally, it is possible to specify recursive dates using an arbitrary formula, using the `function` declaration in a `.cld` file. The use of the `function` declaration is beyond the scope of this manual, but interested designers might want to take a look at the documentation contained in `dates.dtx` for a discussion of this declaration, and an example that generates a note of the phases of the moon.

4 Calendar, Class and Package Options

In this section, all of the options to all of the styles are explained. "Options" come in two flavors. One is accepted by the calendar package or class that makes a given calendar environment available. Such options are fed to the `\usepackage` or `\documentclass` commands in the conventional way, through a square-braced argument. These are referred to here as *class or package options*. The second type of option is given in the second argument to the calendar environment itself, and are referred to here as *calendar options*.[2] Calendar options are given as a comma-delimited list. *Boolean* calendar options take effect when they appear by themselves in the argument text. *Variable* calendar options must be followed by an equal sign and some value, possibly in curly braces. Figure 16 gives examples of each type. To conclude our explanatory tour, we will examine the options recognized by each style in the bundle. If you would like to see samples of any of the styles, please extract and print the appropriate demo file from the distribution.

4.1 Timesheets

The timesheet style produces a timesheet for every date in the controlling range. Timesheets have a column of time blocks marked out on the left side of the page, a list of appointments (if any) to the right of that, and a blank area for recording actions taken (or doodling pictures) on the far right of the page.

4.1.1 Package Options

This style is provided as a package. Its only options are language options declared in `dates.cfg`.

[2] Calendar options are processed using the `keyval` package. See the documentation of that package if you would like more detailed information about how these work.

```
\documentclass{timetabl}
\begin{document}
\begin{timetable}
  {notimes,
   title={Law Department Timetable},
   labels={Monday,Tuesday,Wednesday,Thursday,Friday},
   start=9:00,
   end=21:00,
   blockminutes=60}
  {}
  Jan 5 1997 to Jan 9 1997
\end{timetable}
\end{document}
```

Figure 16: Examples of boolean and variable options

4.1.2 Calendar Options

The following formatting options are supported by the `timesheet` environment:

`title` The string fed to this option will form the title of the timesheet. Robust commands (like `\LaTeXe`) should be safe here. The default string is `Timesheet`.

`leftright` This option can be used to alter the proportion of space taken up by the right and left columns. The syntax is a pair of integers separated by a slash. The default value is `1/1`, for evenly spaced columns. The values express a ratio, so you do not need to make the values add up to anything in particular.

`start` This is the starting time of the timesheet. The format is colon-delimited 24-hour time (i.e. `17:00` for five o'clock in the afternoon). The default starting time is `8:00`.

`end` Like `start`, but specifies an ending time for the timesheet. The default ending time is `17:00`.

`blockminutes` If you want the timesheet to be split into time blocks of equal duration, you can use this option to select the length of the blocks in minutes. This must divide evenly (without a remainder) into the span of time specified by the start and end of the timesheet. The default value of this option is `15`, for fifteen-minute blocks.

`blockgroup` A line will be drawn all the way across the timesheet at the end of every group of time blocks. The number of blocks in a group can be set using this option. The default is `4`, which puts a line at every hour boundary with the default value of `blockminutes`.

`width` If you don't want the timesheet to fill the whole width of the page, you can use this option to specify a smaller size. The feedback given when the style is run gives you the size in points for that run of LaTeX, which may be useful in making formatting adjustments.

`leftspace` This can be used to specify an explicit left-offset value. The default is `\fill`.

`rightspace` Like `leftspace`, but for the right side of the timesheet. The default is `\fill`, for a centred table.

`titletype` This sets the default typeface for the overall title of the timesheet.

`labeltype` This sets the default typeface for the labels (Plan and Action) at the top of the sheet.

`timelabeltype` This sets the default typeface for the time markings on the lefthand side of the page.

`timeitemtype` This sets the default typeface for the times associated with appointments.

`itemtype` This sets the default typeface for the item text itself.

4.2 Weekly

The `weekly` style prints weekly calendars for use with a Filofax(tm) or other personal planner. Pages are framed with cut-lines, and punch-marks show where to cut holes at the edge of the page. For each date, the style initially attempts to set the event and date text as a single column. If the text overflows the box for that date, the style will attempt to recover by resetting the text as two columns.

4.2.1 Class Options

This style is provided as a LaTeX class file. It has the following special features and limitations:

- The text of the calendar is always rotated, using the `lscape` package from the `graphics` bundle. As a result, calendars generated using this package can only be printed using `dvips` or other PostScript(TM) print drivers supported by the `graphics` bundle.
- If the `graphics` bundle has not been configured for your system, you can specify the driver and other options for its use by giving LaTeX options to the class when it is loaded.
- While the paper size can be specified using options like `a4paper`, this will have no effect; the actual paper size is governed by calendar options.
- Documents created with this style should contain nothing other than a calendar environment.

4.2.2 Calendar Options

The following formatting options are supported by the `weekly` environment:

`firstday` This is the starting day of the calendar, expressed as an integer. Friday (the default) is "`5`".

`punchcluster` Most filofaxes have two or or more clusters or groups of punches to hold the pages in place. This option sets the number of punches in each cluster. The default is three.

`intraspace` This governs the space between punchouts within a group. The default is 19.25mm.

`punchgroups` This option sets the number of groups of punches. The default is two groups.

`interspace` This option sets the distance between the groups of punches. The default is 51.25mm.

`pageheight` This fixes the height of a filofax page (not the physical paper on which it is printed). The default is 172mm.

`pagewidth` This fixes the width of an individual page. The physical printed area will be twice this figure. The default is 95mm, for a 190mm printed area.

`grip` This adjusts the distance from the edge of the page to the outer edge of the punchouts. Default is 5mm.

`punchmargin` This adjusts the distance from the edge of the text to the inner edge of the punchouts. Default is 2mm.

`punchsize` Size, in points, of punchouts. Default is 15.

`topspace` Gap between top of filofax page and top of text page.

`bottomspace` Gap between bottom of filofax page and bottom of text page.

`jawspread` If set to a positive length, this places a set of rules the width of the punchmarks on either side of each punchhole, centered on its center and spread the distance specified. This can be useful as a guide with some one-hole punches that are designed to be used "blind".

`jawline` Sets the width of the lines used to make jawmarks. Default value is `0.4pt`.

4.3 Monthly

The `monthly` style produces a simple monthly calendar, similar to the ones you can buy in shops. The LaTeX Calendar version does not include cartoons by Gary Larson or pictures of the latest fave rave teen band (at least by default), but you do have access to all of the goodies available in the other packages in the bundle (color coding, et cetera). Ordinarily you will want to print this using the `lscape` package and its `landscape` environment.

4.3.1 Package Options

This style is provided as a package. It only recognizes language options declared in `dates.cfg`.

4.3.2 Calendar Options

`title` The string fed to this option will form the title of the calendar. Robust commands (like "LaTeX 2ε") should be safe here. The default is the name of the month, followed by the year.

`labels` This is an optional comma-delimited, brace-enclosed string giving a list of seven labels to be used for days of the week. This was once used to make calendars in languages other than English.

`firstday` This is the starting day of the calendar, expressed as an integer. Sunday (the default) is "0".

`titletype` This sets the typeface for the title. The default is `\large\bfseries`.

`daynametype` This sets the typeface for the day-name labels. The default is `\bfseries`.

`datetype` This sets the default typeface for the day in each box. The default is `\bfseries`.

`texttype` This sets the default typeface for the entry texts. The default is `\tiny`.

4.4 Yearly

This produces a simple yearly calendar. It was the first of the Calendar style packages to be written, and in some ways it is the most unsatisfactory; I invite others to use this style as a model and improve on it. But for workaday needs — to produce a simple one-year calendar on a single piece of paper — it works well enough.

4.4.1 Class or Package Options

This is distributed as a package.

4.4.2 Calendar Options

The following options are recognized by the `yearly` style:

`title` The string fed to this option will form the title of the calendar. Robust commands (like "LaTeX 2ε") should be safe here. The default is the number of the year.

`labels` This is an optional comma-delimited, brace-enclosed string giving a list of seven labels to be used for days of the week. This can be used to make a calendar in a language other than English or German.

`firstday` This is the starting day of the calendar, expressed as an integer. Sunday (the default) is "0".

4.5 Timetable

This package organizes date items in a format suitable for conference schedules, itineraries, academic teaching timetables and the like. It is a versatile package, worth playing around with if you want to use this bundle in an institutional context.

4.5.1 Package Options

This style is provided as a package. Its only options are language options declared in `dates.cfg`.

4.5.2 Calendar Options

The options valid for use with the `timetable` environment are:

`leftspace` Amount of space to the left of the table (default is `\hfill`, but with table sized to exactly fit the margins).

`rightspace` Amount of space to the right of the table (default is `\hfill`, but with table sized to exactly fit the margins).

`width` Width of the table. The default is `\textwidth`.

`title` The string fed to this option will form the title of the timetable. Robust commands (like "LaTeX 2_ε") should be safe here. The default is "Conference Schedule".

`notimes` If this option is given, the time of each item is not given in the body of the timetable. If all entries begin and end exactly on the boundaries arrived at by applying `blockminutes` to the range of the table, no information is lost, and the table will have a cleaner appearance and be easier to read.

`start` Time from which schedule should begin, with hours and minutes separated by a colon. The default is `8:00`.

`end` Time at which the schedule should end. The default is `17:00`.

`blockminutes` The number of minutes in each time block. This should divide *evenly* into the total number of minutes within the range specified by `start` and `end`. The default is 60.

`blocks` As an alternative to `blockminutes`, you can specify the precise periods of time you want included in each time block. The format is a set of comma-delimited time ranges, all enclosed in a pair of matching braces. For example:

```
blocks={8:00-9:00,9:00-10:00,4:00-5:00}
```

`labels` This is an optional comma-delimited, brace-enclosed string giving a list of labels to be used for the day headings.

`titleface` The typeface used for the title.

`labelface` The typeface used for the day labels.

`timelabelface` The typeface used for the time labels on the left edge of the table.

`timeitemface` The typeface used for the time of each entry (suppressed altogether if the `notimes` option is given).

`itemface` The typeface used for the entry text.

4.6 Event List

This is a simple package that produces a list of events in cronological order. It is not particularly elegant, and users are invited to improve on this model.

4.6.1 Class or Package Options

The only options recognized by this package are language options declared via `dates.cfg`.

4.6.2 Calendar Options

This package does not recognize any calendar options at present.

4.7 HTML Monthly Calendar

This package produces a monthly calendar in HTML. The calendar does not take advantage of any table features of HTML, and can be viewed on any browser, including the Lynx browser for character terminals.

4.7.1 Class or Package Options

The only options recognized by this package are language options declared via `dates.cfg`.

4.7.2 Calendar Options

This package recognizes he following calendar options:

`title` The string fed to this option will form the title of the calendar. The default is the name of the month, followed by the year.

`firstday` This is the starting day of the calendar, expressed as an integer. Sunday (the default) is "0".

`outputfile` This is the name of a file to which the HTML output should be written.

4.8 HTML Event List

This is a simple package that produces an HTML list of events in cronological order.

4.8.1 Class or Package Options

The only options recognized by this package are language options declared via `dates.cfg`.

4.8.2 Calendar Options

The only option recognized by this package is `outputfile`, which sets the name of the file to which the HTML output should be written.

4.9 HTML Month and Event List

There are CGI scripts that do the same thing as this script — produce one or more monthly calendars, with jump-links on appropriate days into a list of events. The difference is that this style offers multi-lingual support, and could be made to work with color by a wizard.

4.9.1 Class or Package Options

The only options recognized by this package are language options declared via `dates.cfg`.

4.9.2 Calendar Options

This package recognizes the same calendar options as the HTML Monthly style.