The rmpage package
Alpha documentation—trust nothing!

Rowland McDonnell
rebecca@astrid.u-net.com

Some time ago

Abstract

The `rmpage` package lets you change page layout parameters in small steps over a range of values using options. It can set `\textwidth` appropriately for the main fount, and ensure that the text fits inside the printable area of a printer.

An `rmpage`-formatted document can be typeset identically without `rmpage` after a single cut and paste operation. Local configuration can set defaults: for all documents; or by class, by printer, and by paper size.

The geometry package is better if you want to set layout parameters to particular measurements.

Contents

1 Introduction **5**
- 1.1 LaTeX's standard classes . . . 6
- 1.2 How does `rmpage` help? . . . 7
- 1.3 Installing `rmpage` . . . 8
- 1.4 Compatibility . . . 8
- 1.5 What made me write `rmpage` . . . 9
- 1.6 Future plans . . . 10

2 Using `rmpage` **11**
- 2.1 `rmpage`'s view of a page . . . 11
- 2.2 Parameter naming conventions . . . 12
- 2.3 Don't read me . . . 12
- 2.4 Option naming conventions . . . 13
- 2.5 Changing the page layout . . . 13
  - 2.5.1 Width of the main text block . . . 14
  - 2.5.2 Length of the main text block . . . 15
  - 2.5.3 Headers and footers . . . 16
  - 2.5.4 Position of the main text block . . . 17
  - 2.5.5 Marginal paragraphs . . . 19
  - 2.5.6 Multiple columns . . . 20
  - 2.5.7 Paper size . . . 22
  - 2.5.8 Founts . . . 23
  - 2.5.9 Printers . . . 25
  - 2.5.10 Date format . . . 25
  - 2.5.11 The `beton` package . . . 25
  - 2.5.12 Typesetting parameters . . . 27

3 Old Using `rmpage` **28**
- 3.1 Some options . . . 28
- 3.2 Some other options . . . 30
- 3.3 Layout details . . . 30
- 3.4 Paper sizes . . . 30
- 3.5 Setting the body text size . . . 32
- 3.6 Founts- . . . 33
- 3.7 Marginal paragraphs- . . . 33
- 3.8 The configuration files . . . 34
- 3.9 Sending `rmpage`-formatted documents elsewhere . . . 36
- 3.10 Speed and what to do about it . . . 37

4 A brief lecture on typography **40**
4.1 Introduction . . . 40
4.2 Positioning the text body . . . 41
4.3 Size of the text body . . . 42
4.4 Typefaces . . . 43

5 All the options (rmpage v0.69 and rmplocal.cfg v0.11) **45**
5.1 Options in `rmpage` . . . 45
5.1.1 Reporting dimensions and tracing calculations . . . 45
5.1.2 Paper sizes . . . 46
5.1.3 Typesetting tightness . . . 48
5.1.4 Textheight setting . . . 49
5.1.5 Headers and footers . . . 49
5.1.6 Columnsep . . . 50
5.1.7 Width of the text body . . . 51
5.1.8 Width setting control . . . 52
5.1.9 Margins . . . 52
5.1.10 Number of columns . . . 54
5.1.11 Paper orientation . . . 54
5.1.12 Headers and footers . . . 54
5.1.13 Positioning the text body vertically . . . 55
5.1.14 Changing the date format . . . 55
5.1.15 Dealing with the beton package . . . 55
5.2 From the configuration file . . . 56
5.2.1 Other paper sizes . . . 56
5.2.2 Marginal paragraph options . . . 59
5.2.3 Touch options . . . 61
5.2.4 More length options . . . 62
5.2.5 Number of columns . . . 62
5.2.6 Loading founts . . . 63
5.2.7 Stuff for beton support . . . 66
5.2.8 Other synonyms for some options . . . 66
5.2.9 Margin options . . . 66
5.2.10 Printer options . . . 66
5.2.11 Rowland's curious options . . . 69

6 How things work **71**
6.1 `\textheight` calculation . . . 71
6.2 `\textwidth` calculation . . . 73
6.3 Hooks . . . 75
6.4 Marginal paragraphs . . . 76
6.5 Dealing with different classes . . . 76
6.6 Different paper types and printers . . . 76
6.7 Headers, footers, and marginal paragraphs . . . 77

7 Configuring `rmpage` 79
7.1 Setting up a new installation 79
7.2 Configuration basics 79
7.2.1 Unknown option error 80
7.2.2 Default options 80
7.3 Configuring `rmpage` for particular classes 81
7.4 Defining a new printer type 81
7.5 Dealing with particular combinations of printer and paper . . . 81
7.6 Telling `rmpage` about a new class 81
7.6.1 Dealing with options 82
7.6.2 Things you can do with your new class number 83

Chapter 1

Introduction

This documentation needs re-writing and tidying up, and `rmpage` needs finishing, but the package is 99% finished even if the documentation's a mess – I'm not happy with it but it's probably usable, so here it is.

The rmpage package has five files: `rmpage.tex` (documentation), `rmpage.sty` (the package file itself), `rmplocal.gfc` (a configuration file), and `rmpgen.cfg` (another configuration file), and `readme`.

Only `rmpage.sty` and `rmpgen.cfg` *need* to be on your TEX search path – read on to find out more about configuring rmpage.

I seem to have ground to a halt on this package – it's worked fairly well for some time and I don't seem to have been able to get things together to sort out the documentation and stuff. I've come across one bug only in the last year, so I thought the best idea would be to upload the package to CTAN so someone else could use it, and perhaps any feedback I get might persuade me to pull my finger out and tidy it all up.

The `dtx` files aren't ready to be typeset yet—this is the only LATEXable documentation.

What I do with `rmpage` in the future depends mainly on what you tell me: if you've used `rmpage` (or decided not to), I would consider it a great kindness if you told me why. I'm also interested in what you like and dislike, any suggestions you have, and anything else about this package and its documentation—an email message just saying '`rmpage` is rubbish, geometry's much better for...' would be useful if that's what you think.

The chapters on how things work and all the options still need a lot of work, and the chapter on configuration isn't how I'd like it to be. This document will eventually be finished and included in a proper `.dtx` file. I thought releasing this package now was best, because I've suddenly become employed, and this final polishing will take quite a long time.

`rmpage` sets LATEX page layout parameters to user-controlled values, without the user having to deal with particular measurements, check whether the result will fit inside the printing area of the selected printer, and so on. This is done with options like: `wider`, `noheaders`, `lower`, and `morecolsep`; all changable layout parameters can be varied in small steps over a range of values.

`rmpage` only changes parameters like `\textwidth` and `\columnsep`: those lengths that affect where the text goes on the page and how much of the page it

occupies. It doesn't change layout parameters that affect the internal appear of your text, such as paragraph indents, spacing around section headings and the like. One of the design aims was to make changing page layout similar to a wysiwyg word processor, where you can use the mouse to fiddle with the page layout, making things a bit bigger and smaller until it's just right.

There's a configuration file for you to play with and hooks galore: `rmpage` is meant to be configured the way you want it—if you normally use A4 paper and produce most documents without headers, you can configure `rmpage` to give you that by default, over-ridable by passing options from your document. If you follow the instructions, local configuration won't stop you producing documents with a modified layout that typeset identically on different systems—you can even copy the modified layout data into your document, and it'll typeset identically on a LaTeX installation without `rmpage`.

The package is meant to be used directly in LaTeX documents, and for creating local classes. I have used it, for example, to create a class for producing theses according to the regulations. I can play about with the layout as much as I like, because `rmpage` will ensure that the final document is within specification—some of the code that does this can be seen in the configuration file.

Aside: this documentation was hell to write—I hadn't realized what a monster this package was until I came to document it. With a bit of luck, you'll be able to use `rmpage` as your flexible friend like I do. I never write a LaTeX document without it. And strangely, even though `rmpage` has been over a year in the making, and has been used constantly in that time, I spotted lots of improvements that needed doing and bugs that needed removing while I was writing this documentation, which I didn't start seriously until I thought I'd finished the first release version of the package. I have a suspicion that writing detailed documentation of software is a very, very useful part of getting it right, especially when the software's moderately complicated. Personally, I'm now suspicious of anything that isn't documented thoroughly; I've looked at the various packages I've got from CTAN and use myself, and I find that those with thorough, clear documentation of what they do *and* how they do it are the ones that seem most usesful and flexible. The poorer-documented packages appear to be less well thought out and less able to do things my way, rather than the author's. By the way, when I refer to an explanation of how something works, I'm not talking about the annotated code with added jargon that the LaTeX core appears to be turning into. This is fairly useful and probably inevitable, but people like me who aren't expert hackers can't understand it and can't find out how to understand it, which is worse.

1.1 LaTeX's standard classes

LaTeX's standard classes have all been designed well, and the LaTeX 2_ε versions have a cunning way of calculating `\textwidth` and `\textheight` that adapts the page layout to the size of paper, which is useful for people like me who never print on US letter paper.

But the standard LaTeX page layout was intended for paper sizes near US letter, and assumes you'll be using headers and footers. If you're not, things

begin to look a bit off; your printed pages have a larger white space at the top than at the bottom, which looks 'bottom heavy'—standard typographical design has the larger gap at the bottom rather than the top, which looks better to the eye, even if only because we're used to it. Personally, I rarely use headers, which means the standard LaTeX classes produce an unpleasant output most of the time. And for some reason, the standard classes don't allow enough space in the header box for type sizes bigger than 10pt. This is one of life's inexplicable mysteries.

If you have different sized margins, the standard classes always make the outside margin the larger one, and the margins are in fixed proportions to each other. This is fine for conventional typeography, but not so good if you're formatting things to go in a ring-binder, for example. Some flexibility in this matter would be good.

And there's the matter of the fixed text height and width of the standard classes. Now then, this is a good idea in one respect, because the user doesn't get to typeset lines that are too long or too short without having to do a bit of work. On the other hand, I for one have often LaTeXed a letter which has *just* run on to two pages; being able to extend the page a small amount would be good under these circumstances. Admittedly, LaTeX 2_ε introduced the `\enlargethispage` command which can help out, but if you make a page more than one or two lines longer this way, the results look a bit iffy—this command only extends `\textheight`, which reduces the size of the bottom margin, and can make the page look bottom-heavy.

A more subtle problem with the standard fixed widths is that one factor affecting the ease of reading is the line width in characters, not in inches. So if you're using a fount with a different number of characters per inch, you can end up with a line that is noticably too long (Zapf Chancery), or too short (Lucida Casual).

1.2 How does rmpage help?

`rmpage` has options (the details come later) to format a page for typesetting with or without headers or footers, and always leaves enough space for a `\normalsize` line of text in the header box. You can specify which margin you want to be the larger one, and adjust the relative proportions of the inside and outside margins. And there are options to change: the width and height of the text taking into account the size of the main text fount; the position of the text on the page (up and down, and left and right); space between columns, above footers, below headers; size and position of marginal paragraphs; and lots, lots, more, so hurry! Buy now while the sale's still on!

`rmpage` isn't like the geometry package: you rarely work with measurements. `rmpage`'s options are of the form: `wider`/`narrower` or `moreheadsep`/`lessheadsep`, and they scale sizes up and down over a large range in fairly small steps if you want. And unlike the Koma-Script bundle of packages, you're not limited to a fixed aspect ratio printing area. Not that there's anything wrong with Koma-Script or geometry: they do a different job.

`rmpage` can take into account different founts, number of columns, and all sorts of things, including the physical printing area of your printer.

Options exist to change all (I think I got them all) of LaTeX's basic page layout parameters, and some new parameters that I created for: setting the size of marginal paragraphs, and changing the position of the text area in relation to the paper area. There are some LaTeX page layout parameters that you don't get to affect directly; this is because of the way I looked at page layout when I wrote `rmpage`. These include `\evesidemargin`, `\topmargin`, and some others, which are fine for a computer assembling a page, but not so good for me, trying to describe a layout I want in terms I like.

I have paramaterized everything that didn't run away, so you can, for example, write a thesis class that limits the text area as specified in the regulations, but still allows the user some flexibility if it's needed.

And `rmpage` knows about a lot more paper sizes, including envelopes and ISO long sizes, which it attempts to handle in an intelligent fashion (it knows one's likely to want to print out 1/3 A3 on A4 paper, for example).

1.3 Installing `rmpage`

You can install `rmpage` by running LaTeX on the file `rmpage.ins`. The `dtx` files might be useful if you want to dig around inside `rmpage`, otherwise throw them away with the `ins` files. Put the `sty`, `pko`, `cfg`, and `gfc` files somewhere in your TeX search path.

The file `rmpage.sty` is the package you call from your LaTeX document, and the `cfg` files contains most of the options and locally-configurable things. Please don't change any of these files—make a copy called `rmplocal.cfg` of either: `rmpgen.cfg` or `rmplocal.gfc`, and change that instead. The file `rmplocal.gfc` is the same as `rmpgen.cfg` with some options commented out to make it faster.

`rmpage` is meant to be configured to suit you; I suggest that everyone changes the default options in `rmplocal.cfg`. Don't do this just yet unless you really want to—have a read of chapter 2 and section 7.1 first.

1.4 Compatibility

`rmpage` has been tested with the June 1996 release of LaTeX. It appears to work well with the `article`, `report`, `letter`, `book`, and `slides` classes, and a number of local classes based on these. `rmpage` doesn't work well with the `ltxdoc` class, because of what `ltxdoc` does with marginal paragraphs. Mind you, `ltxdoc` doesn't seem to have much success with marginal paragraphs anyway. `rmpage` seems happy with the `ltxguide` class, but I've not tested it thoroughly.

Because `rmpage` only changes LaTeX parameters at the beginning of a LaTeX run, it doesn't in general have trouble working with other packages and classes as long as `rmpage` is loaded last. `rmpage` includes code to help it work with the PSNFSS packages, the beton package, and the foils class. The beton package needed a little extra work because it changes `\baselineskip` after all packages have been loaded; foils uses four (rather than three) extra-large base point sizes, which again took a little extra code. The PSNFSS package support is

just for convenience, so you can load a fount with one option rather than a `\usepackage` command and an option to `rmpage`.

In general, if `rmpage` is loaded after the document's normal size fount has been selected, and after the document class has finished setting the various text layout parameters, there should be no problems—. If you are combining `rmpage` with a package that also changes page layout parameters, you will have to find out how both packages work to ensure you get what you want. Loading `rmpage` last is usually enough to ensure everything works right. For example, `rmpage` must be loaded after `setspace` has been used to set the line spacing for a document, so that `rmpage` can set `\textheight` to a valid value.

> Two things to watch for are changes to the main document fount, and changes to `\baselineskip`. `rmpage` calculates `\textheight` as an integer multiple of `\baselineskip` plus `\topskip`. If these are changed after `rmpage` has been called, you'll probably have lots of bad page breaks. `\textwidth` is normally taken to be a certain number of average-sized characters; if `rmpage` has a false idea about the typeface and size you are using, `\textwidth` will probably not be set appropriately.
>
> If you ensure that normal size in the normal body fount with the normal `\baselineskip` has been selected before loading `rmpage`, everything should always be fine. The `beton` package sets `\baselineskip` `\AtBeginDocument`; other packages and classes which do this kind of thing will almost certainly need attention to get `rmpage` to work right.

1.5 What made me write `rmpage`

I started using LaTeX 2.09 about eight or nine years ago when I was an undergraduate ('ere, Colin, 'ave you got a word processor on that bloody great workstation of yours? No, but I've got something better...) and quickly realised that the standard formats looked daft on A4 paper. No worries, there was this `a4l` style file that sort of sorted things out. After a bit, I wanted slightly wider columns, sometimes centred on the page and sometimes not. I found out how to write style files and eventually I ended up with a suite of simple style files that let you fiddle with the printing area by selecting one file for the kind of text area you wanted: centred, not centred, wide, very wide, long, standard LaTeX length, my standard length, all hard-coded to A4 paper with no headers.

I was chewing over the idea of introducing the idea of paper sizes myself, so I could write style files that weren't hard-coded to any particular paper size, when LaTeX 2_ε came out with the job already done, and packages you could pass options to and all that good stuff.

So what I did was use my original LaTeX 2.09 packages to form the basis of the original `rmpage`, which let you do everything the original styles did, but all in one file instead of over a dozen, and let you format your text on any size paper with or without headers or footers. I decided what I really wanted was a package that gave me wysiwyg-style flexibility (you know, the way you can extend line length just a bit with the mouse, without worrying about the actual numbers) without producing poor layouts, and taking advantage of LaTeX's 'extensive macro capability'[1] to build in a bit of intelligence.

[1] Whoever first used this phrase should be shot, after the politicians but before the lawyers

I looked at the LaTeX 2_ε way of calculating the text region, and used those ideas in my package, and started adding options to change more aspects of the printing area. And more, and more, and more. The result was a mess that could change almost anything, and supported any size paper, different printers, and so.

The transmogrification from the original large mess to the current large mess was in two main steps: I started out by tidying up bits of code piecemeal, rationalized command names and the like, and tried to work out how everything worked together. I'd done as much of this as I could, then started to document `rmpage` systematically for myself, making notes on what I intended to change when I'd finished documenting the package.

Eventually I got fed up, and decided that a good spring cleaning was in order. I read some typeography books, some British Standards (which are mainly ISO standards too, so I'm not being parochial) looked at the koma-script, vmargin, and geometry packages, and cleaned up the code good and proper, none of this pussy-footing around with careful plans. I parameterized some things which had escaped the first time round, changed the numbers to give a rational, æsthetically pleasing, and functional spread of values for everything[2], and got the whole thing more-or-less sorted, with a few extra bits thrown in where they were missing. The result seems much more useful that my original careful plan would have produced, so I'm happy. Writing the documentation has smoothed out several things I wasn't very happy with to begin with, improved some features, added some others, and unearthed more bugs than was reasonable given that I'd tested the bloody thing, honest.

1.6 Future plans

What I do with `rmpage` depends mainly on what you tell me; if you've used `rmpage` (or decided not to), I would consider it a great kindness if you let me know what you think about it—two words or two pages: whatever you might tell me would be useful and appreciated. I'm interested in what you like and dislike, any suggestions you have, and anything else about this package and its documentation, gonzo ontology, the books of Harlan Ellison and Robert Anton Wilson, good beer, fast bikes, and prawn crackers, but don't worry about most of that.

I intend to make the `rmpage` code more elegant, and make `rmpage` more *useful*—if you think of any useful changes or additions, please let me know. Of course I'll try to fix any bugs and misfeatures that you report.

I'm working on reducing the restrictions on the use of options—I hope to arrange things so that more options can be used in an `\ExecuteOptions` statement, and things like that.

A project that I'll finish eventually is a version of `rmpage` that can be processed with doc and docstrip in the conventional way, but don't hold your breath—it's taken me perhaps two years to get this far. But I will fix bugs quicker than that.

[2] That's my story, and I'm sticking to it.

Chapter 2

Using rmpage

Section 2.5 on page 13 of this chapter describes how to use `rmpage` to change the page layout. The sections before that are background information intended to explain the jargon and conventions I use in this document, and a little about the philosophy behind `rmpage`—all of which should make the rest of this an easier read.

`rmpage` is a LaTeX package written to change the page layout. You can control it with options, and by editing a configuration file. You don't need to edit the configuration file for `rmpage` to be useful, but doing so can save you time and effort. I suggest you read this chapter and play with `rmpage` a little, then read chapter 7 on page 79 to find out about local configuration.

`rmpage` is a normal LaTeX package, so put:

```
\usepackage{rmpage}
```

in the preamble of your document.

You can pass options directly to `rmpage` in the optional argument of the `\usepackage` command, but I almost always put all my options in the optional argument of the `\documentclass` command. This is because `rmpage` uses several standard options; if I made a habit of passing options to `rmpage` directly, I might forget that the document class needs to know about (for example) the `twocolumn` option.

`rmpage` was designed to change LaTeX's page layout parameters, but it doesn't change all of them in a direct way—read on to see what I mean. If you aren't familiar with LaTeX's basic page layout parameters, have a look at a copy of the LaTeX manual, ask a convenient guru, or use the layout package to show you what they are—I think that what they are is obvious from their names, but I'm not a Finnish LaTeX novice, so my opinion is clearly suspect.

Figure 3.1 on page 31 shows the output from the layout package's `\layout` command. I've fooled the `\layout` command into thinking this is a one-sided document, and this document is formatted without headers, so you can't see clearly that there is a box of height `\headheight` a distance `\headsep` above the main text body. This box contains the header, if one exists.

2.1 rmpage's view of a page

The user interface to `rmpage` takes the view that a page consists of a physical paper size, with a non-printable border around its inside edge. `rmpage` will

not produce a layout that attempts to put ink beyond the printable region thus defined.

`rmpage` considers the *main text block* to consist of the header, body text, and footer. This is the region that `rmpage` moves up and down with `altitude` options, and left to right with `offset` options. The body text is part of the main text block, and is the matter that fits inside the area defined by `\textwidth` and `\textheight`.

Marginal paragraphs stick on the side of the main text block. They begin a certain distance from the side of the body text, and extend to within a certain distance of the edge of the page, or to a certain maximum width.

The space between the main text block and the edge of the paper: top, bottom, left, and right; is considered as four different margins, measured from the edges of the paper. LaTeX's `\evensidemargin`, `\oddsidemargin`, and `\topmargin` parameters measure margins differently, from a point one inch in from the top left hand corner of the paper.

Asking for `noheaders` or `nofooters` reduces the size of the space left for the approriate element to 0 pt—respectively `\headheight` *and* `\headsep`, or `\footskip`, so the main text block might consist of body text only.

2.2 Parameter naming conventions

I refer to various parameters in this document, so you might find it useful to know what the names are supposed to mean.

In general, parameter names ending in `clearance`, `clear`, or `margin` refer to a distance from the edge of the paper. Parameter names containing `sep` refer to a distance between text elements on the page. `mpar` means marginal paragraph; `width` refers to horizontal dimensions; `height` or `length` refer to vertical dimensions.

Parameter names ending in `option` contain numbers that control what value the parameter referred to is set to. Parameter names containing `min` or `max` are minimum or maximum limits for the parameter referred to.

All `rmpage`'s parameters and commands begin with `\RM`; those beginning with `\RM@` are not meant to be set outside a class, package, or configuration file. The one exception to this is the `\sloppiness` command.

2.3 Don't read me

Without any options specified by your document or the configuration file, on paper about A4 or US letter size, `rmpage` will produce a slightly different format to the standard classes: `\textwidth` and the position of the text body on the page will be a fraction of a point different; everything else should be the same (if not, you've found a fault—please let me know). Smaller paper sizes, around A5 or half US letter, will have a noticeable wider `\textwidth`. You can force `rmpage` to make `\textwidth` and `\textheight` identical to the standard values (so line and page breaks are not changed) with the `stdwidth` and `stdlength` options; main text block positioning is never identical to standard.

`rmpage` with the standard configuration file follows LaTeX's defaults and produces layouts very close to standard LaTeX. You can change this by editing

the configuration file—see chapter 7 for the details. I wrote `rmpage` expecting that everyone would edit the configuration file to match their preferences; for example, if you usually don't use headers, or if you usually don't print on US letter paper.

If you are using typefaces other than the standard Computer Modern Roman, or a package that changes `\baselineskip`, have a look at section 1.4 on page 8 and section 2.5 on page 13—there are things that need doing to avoid a poor layout.

The observant will notice that I prefer spelling things according to the Oxford English Dictionary, rather than Webster's. Fear not: I realize that LaTeX follows US English convention, so `rmpage` includes options spelt both ways where there's a difference.

2.4 Option naming conventions

`rmpage`'s options are largely of the form: `narrowest`, `narrower`, `narrowish`, `normalwidth`; or `mostheadsep`, `moreheadsep`, `moreishheadsep`, `normalheadsep`. These two examples are each part of an option set (they continue with `widish` and `lessishheadsep`). Any option without `touch` or `t@uch` in its name is considered a main option.

You should use only one main option from each set at a time—if you do use more than one, `rmpage` will apply the settings of the option that is declared last in the package file.

The `touch` options all step up or down one third of the way (usually in a geometrical sequence) to the next main option. The `t@uch` options are identical, but can only be used in class and package files. The following two examples produce an identical `\textwidth`:

```
\usepackage[wider,t@uchwider,touchwider]{rmpage}
\usepackage[widest,touchnarrower]{rmpage}
```

The following three examples produce a smoothly increasing `\textwidth`:

```
\usepackage[touchwider]{rmpage}
\usepackage[widish,touchnarrower]{rmpage}
\usepackage[widish]{rmpage}
```

It's complete okay, but slightly silly because it has no effect, to combine, say, `touchwider` and `touchnarrower`.

2.5 Changing the page layout

`rmpage` is controlled by options passed to it in the conventional way, and by various things you can do to the configuration file. This section explains the basic use of most of the options. Chapter 5 on page 45 lists all the options and what they do. Chapter 7 on page 79 deals with the configuration file.

2.5.1 Width of the main text block

According to the text books, the optimum width of a block of text is about 1.5–2.5 alphabets in the main fount. This is about 45–75 characters (including spaces and punctuation) or ordinary English prose. When you set the width of the main text block, `rmpage` measures the width of one column, and warns you if it exceeds these limits. Note that the standard width is at the upper limit for optimum readability; any increase will produce a warning. I very strongly suggest you use multiple columns if you find yourself using a width wider than `widish`.

`rmpage` has options for producing multiple column layouts: see section 2.5.6 on page 20 for more details. If you are producing displayed material (a single large table on a page, for example), read the section below, called 'On other width setting controls'.

`rmpage` sets the width of the main text block (the header, body text, and footer) with these options, which are referred to as the width option set:

`widest`, `wider`, `wide`, `widish`,
`normalwidth`,
`narrowest`, `narrower`, `narrow`, `narrowish`.

Make sure you only use one of the above options at a time. The `touch` options can be used with any of the main options; they are often exactly what's needed what used alone: `touchwider` and `touchlonger` have often reduced my document's page count to what I wanted.

The options:

`touchwider` `touchnarrower`
`t@uchwider` `t@uchnarrower`

give a width one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

There are four main width options that pay no attention to the `touch` options. They are:

`stdwidth`	Exactly the width calculated by the standard classes
`fullwidth`	The full width of the printable region
`oneinchmargins`	The left and right margins sum to 2 in—average 1 in
`halfinchmargins`	The left and right margins sum to 1 in—average 0.5 in

The `fullwidth` option fills the width of the printable region as well as it can, ensuring the specified relationship between the inside and outside margins. You will usually get a larger `textwidth` if you also ask for `centre` or `center`; see section 2.5.4 on page 17.

The `one-` and `halfinchmargins` options give inside and outside margins of that measurement only if you are printing centred; otherwise, the average margin size is as specified (e.g., inside 1.2 in, outside 0.8 in; $(1.2 + 0.8)/2 = 1$)

Other width setting controls

The initial `\textwidth` is normally calculated as the smaller of two different widths: one, a certain number of characters; the other, a certain fraction of

the \paperwidth. The precise figures depend on the width options you've used.

This is not always appropriate—for example, if you are producing a weekly timetable on A4 landscape paper, I can't see why rmpage should pay attention to the character-based width. So I created these options:

characterwidthset	Choose the character-based width regardless
paperwidthset	Choose the paper-based width regardless
bothwidthset	Default: choose the smaller of the two widths

Note that rmpage never ignores its paper-based limits: saying characterwidthset will produce a printable layout that takes notice of all the restrictions documented elsewhere.

I've provided a ringbinding option which sets the minimum allowed inside margin to at least 15 mm if you are printing in portrait orientation, and does nothing but warn you if you are using landscape orientation. It's probably not a good idea to use this with long paper sizes, but no check is made.

2.5.2 Length of the main text block

rmpage sets the height of the main text block (the header, body text, and footer) with these options, which are referred to as the length option set:

longest, longer, long, longish,
normallength,
shortest, shorter, short, shortish.

Make sure you only use one of the above options at a time. The touch options can be used with any of the main options; they are often exactly what's needed what used alone: touchwider and touchlonger have often reduced my document's page count to what I wanted.

The options:

touchlonger touchshorter
t@uchlonger t@uchshorter

give a width one third of the way towards the next main option. The t@uch options can only be used in a class or package file.

The height you get is (where Z is an integer; the body text is $Z + 1$ lines long):

$$Z \times \texttt{\baselineskip} + \texttt{\topskip} + \texttt{\headheight} + \texttt{\headsep} + \texttt{\footskip}$$

The length of the main text block is set to be a certain fraction of \paperheight, so \textheight will increase if you turn headers or footers off.

2.5.3 Headers and footers

`rmpage` doesn't select a page style to use or not use headers or footers—you've got to arrange for that to be done separately with a `\pagestyle` command. It does calculate a page layout that does or does not allow space for a header or a footer. If you turn footers off and forget to choose a footer-free page style, the result is mildly comical.

You can allow space (or not) for headers and footers using these options:

`headers` `noheaders`
`footers` `nofooters`

Telling `rmpage` not to allow space for either headers or footers will increase `\textheight`, and vice-versa. See section 2.5.2 for more about this.

LaTeX's standard classes allow a box 12 pt high for headers. This is too small for point sizes greater than 12 pt, so `rmpage` changes the size of the box containing the header to be `\baselineskip`. If you want to use a header which is a different height to that, define the command `\RMheadheight` to be whatever the height is before calling `rmpage`. For example, if your header is to be 32 pt high, do this:

```
\providecommand{\RMheadheight}{32pt}
\usepackage{rmpage}
```

If you don't usually use headers, I suggest that you edit the configuration file so, by default, `rmpage` calculates a page layout that doesn't allow space for them. See chapter 7 on page 79 for how to do this.

These options let you change the space between the header and the body text—use only one of these at a time:

`mostheadsep` `moreheadsep` `moreishheadsep`
`normalheadsep`
`lessishheadsep` `lessheadsep` `leastheadsep`

These options let you change the space between the footer and the body text—use only one of these at a time:

`mostfootskip` `morefootskip` `moreishfootskip`
`normalfootskip`
`lessishfootskip` `lessfootskip` `leastfootskip`

The `footskip` options scale the gap between the *top* of the footer and the bottom of the body text—the calculation assumes that the footer is one line high. The standard LaTeX parameter `\footskip` is the distance from the bottom of the body text to the bottom of the footer.

Both the option sets above have corresponding `touch` options—these can be used with any of the main options above:

`touchmorefootskip` `touchlessfootskip`
`t@uchmorefootskip` `t@uchlessfootskip`
`touchmoreheadsep` `touchlessheadsep`
`t@uchmoreheadsep` `t@uchlessheadsep`

These options increase or decrease the corresponding parameter one third of the way towards the value given by the next main option.

2.5.4 Position of the main text block

Vertical position

You can raise and lower the position of the main text block on the page using the altitude set of options:

```
highest higher high  highish
normalaltitude
lowish  low    lower lowest
```

Be sure you only use one of the main options above at a time. The `touch` options below can be used with any of the main options.

The options:

```
touchhigher touchlower
t@uchhigher t@uchlower
```

give a width one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

The text books say that the white space at the bottom of a page should be larger than the white space at the top. LaTeX's and `rmpage`'s standard setting splits the space evenly between top and bottom; this results in an apparently larger space at the bottom—the printing appears to finish at the bottom of the body text, because the footer is usually just a page number.

The altitude options consider the top margin to be the space above the top of the header box, and the bottom margin to be the space below the footer baseline. They work by changing the ratio between these two spaces; the sum of the top and bottom margins is not changed.

But if you ask for a page layout which would result in text exceeding the various vertical limits, `rmpage` will increase the top or bottom margin as appropriate without attempting to retain a fixed ratio between them. For example, if the layout would extend 2 mm off the top of the printable area, the top margin would be increased by 2 mm and the bottom margin would remain the same.

This is different to the horizontal position options, which do ensure a fixed ratio between the inside and outside margins; if the inside margin is reduced, the outside margin is reduced to retain the requested proportions.

These two ways of doing things were deliberate design decisions; if anyone thinks I've got it wrong, please email me and try to persuade me that you're right.

Horizontal position

The horizontal positioning of the main text block is controlled by three types of options which: vary the ratio between the larger and smaller margins, switch the larger margin from the inside to the outside, and force both margins to be the same size or not.

The options:

`centre` or `center`	Equal inside and outside margins
`notcentre` or `notcenter`	Usually unequal inside and outside margins.

control whether or not the text will be centred horizontally. Using the `centre` option forces the inside and outside margins to be the same; the `notcentre` option means they can be different. Be sure you only use one of these options at a time.

The options:

`stdmargins` Outside margin larger
`notstdmargins` Inside margin larger

control which of the two margins on a page will be the larger when you have requested a `notcentre`d layout. LaTeX's convention, and standard typographical convention, has the larger of the two margins on the outside. This is given by `stdmargins` (an abbreviation for standard margins). The `notstdmargins` option gives the opposite effect—it is not standard practice to have the inside margin larger than the outside margin, although it is useful when you're producing material for a ring binder. Be sure you only use one of these options at a time.

You can shift the main text block from right to left using the `offset` set of options:

`mostoffset` `moreoffset` `moreishoffset`
`normaloffset`
`lessishoffset` `lessoffset` `leastoffset`

These options change the difference between the inside and outside margin: `leastoffset` gives you centred printing, with equal inside and outside margins. `mostoffset` produces inside and outside margins in the proportions 87% : 13%.

These options produce a particular ratio between the inside and outside margins. If you have asked for a very wide `\textwidth` which is limited by the non-printing margin of your printer, `\textwidth` might be reduced to produce margins in the requested proportions. If you have symmetrical left and right non-printing margins on your printer, you can only completely fill the available width if you request centred printing with the `centre` or `leastoffset` options.

Be sure you only use one of the main options above at a time. The `touch` options below can be used with any of the main options.

The options:

`touchmoreoffset` `touchlessoffset`
`t@uchmoreoffset` `t@uchlessoffset`

give an offset one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

An oddity to watch out for is this: `leastoffset` produces the least offset between the left and right margins, namely none. If you use the `leastoffset` and the `touchlessoffset` options together, you get what I call a negative offset. That is, if you've asked for `stdmargins`, you'll end up with a larger inside margin than outside, and if you've asked for `notstdmargins`, you get a larger outside margin than inside. `rmpage` will draw your attention to this is it happens.

2.5.5 Marginal paragraphs

Marginal paragraphs begin a distance `\marginparsep` away from the side of the main text block, and extend to a distance `\RM@mparclearance` from the edge of the paper. The maximum width of a marginal paragraph is given by `\RM@maxmparwidth`.

With a conventional layout on paper similar to A4, marginal paragraphs usually fill the space from `\marginparsep` (about 4 mm) away from the main text block, to `\RM@mparclearance` (about 10 mm) in from the edge of the paper. `\RM@maxmparwidth` (about 50 mm) is not usually a limit.

All of these parameters can be controlled by options. `\marginparsep` is a standard LaTeX length which is initially set by the class file; the other two are `rmpage` commands, and are initially set to a fraction of `\paperwidth`.

The options:

`mostmparsep` `moremparsep` `moreishmparsep`
`normalmparsep`
`leastmparsep` `lessmparsep` `lessishmparsep`

control the size of the gap between the marginal paragraph and the main text block.

Be sure you only use one of the main options above at a time. The `touch` options below can be used with any of the main options.

The options:

`touchmoremparsep` `touchlessmparsep`
`t@uchmoremparsep` `t@uchlessmparsep`

give an offset one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

The options:

`mostmparclearance` `moremparclearance` `moreishmparclearance`
`normalmparclearance`
`lessmparclearance` `lessishmparclearance` `leastmparclearance`

control the size of the gap between the marginal paragraph and the edge of the paper.

Be sure you only use one of the main options above at a time. The `touch` options below can be used with any of the main options.

The options:

`touchmoremparclearance` `touchlessmparclearance`
`t@uchmoremparclearance` `t@uchlessmparclearance`

give an offset one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

The options:

`mostmaxmparwidth` `moremaxmparwidth` `moreishmaxmparwidth`
`normalmaxmparwidth`
`lessmaxmparwidth` `lessishmaxmparwidth` `leastmaxmparwidth`

control the maximum size of marginal paragraphs; marginal paragraphs usually stop `\RM@mparclearance` away from the edge of the paper, but there is the additional limit that marginal paragraphs cannot be larger than RM@maxmparwidth.

Be sure you only use one of the main options above at a time. The `touch` options below can be used with any of the main options.

The options:

`touchmoremaxmparwidth` `touchlessmaxmparwidth`
`t@uchmoremaxmparwidth` `t@uchlessmaxmparwidth`

give an offset one third of the way towards the next main option. The `t@uch` options can only be used in a class or package file.

If you want an unusual layout with a very much larger than usual maximum marginal paragraph width, or a very much larger than usual gap between the marginal paragraph and the edge of the paper, the `largebase` options will double the initial size of these parameters. This means that all the sizes produced by the corresponding options above are doubled. The `normalbase` options give the default size.

The additional options:

`normalbasemaxmparwidth`	Normal maximum size marginal paragraphs
`largebasemaxmparwidth`	Double sized maximum size marginal paragraphs
`normalbasemparclear`	Normal gap between the edge of the paper and the end of marginal paragraphs.
`largebasemparclear`	Double sized gap between the edge of the paper and the end of marginal paragraphs.

are intended to be used only when the range of sizes given by the conventional options aren't enough; `more` and `touchmore` combine to double the size of any of the marginal paragraph parameters (the precise factor is 2.0394).

2.5.6 Multiple columns

There are two things to consider with a multiple column layout: the text columns themselves, and the gap in between.

The width of text columns

If you are producing a layout with more than one column, `rmpage` usually needs to know how many columns, because it takes into account the width of each text column, measured against the width of the average character.

If you're using LaTeX's standard `onecolumn` or `twocolumn` options, `rmpage` takes note:

`onecolumn` Default. Produce a layout assuming one text column
`twocolumn` Produce a layout assuming two text columns

You must pass these options to the class file by placing them in the optional argument to the `\documentclass` command, or your text will not be set in the number of columns you expect.

You might be producing a multiple column layout using the multicol package. If so, you should use different options passed to `rmpage` to tell it how many text columns your document will be set in. The following options—`onecolumnwidth` is the default—tell `rmpage` to calculate a layout assuming the text body will be set in the named number of text columns:

`tencolumnwidth ninecolumnwidth eightcolumnwidth sevencolumnwidth sixcolumnwidth fivecolumnwidth fourcolumnwidth threecolumnwidth twocolumnwidth onecolumnwidth`

Make sure you only use one of the twelve options above at a time.

If you are producing a document with different numbers of columns in different places, try starting out by telling `rmpage` that you are using the smallest number of columns in your document. For example, if your document has three columns in some places and four columns in others, pass the `threecolumnwidth` option to `rmpage`. If the width needs changing after that, begin by trying the `width` options in section 2.5.1 on page 14.

The space between columns

The main options:

`mostcolsep` `morecolsep` `moreishcolsep`
`normalcolsep`
`lessishcolsep` `lesscolsep` `leastcolsep`

increase or decrease the separation between the columns—they scale the standard LaTeX `\columnsep` parameter. The default `normalcolsep` option does nothing—you get LaTeX's standard column separation. Make sure you only use one of these main options above at a time.

The `touch` options below are meant to be used with any of the main options, and increase or decrease the gap between the columns one third of the way towards the next main option.

`touchmorecolsep` `touchlesscolsep`
`t@uchmorecolsep` `t@uchlesscolsep`

If you don't like LaTeX's standard `\columnsep`, the options below calculate a different default value:

`adaptivecolumnsep`	Calculates a normal `\columnsep` which is 2.3 times the average character width of the selected fount—a 0.1 pt increase for 10 pt Computer Modern Roman; quite a bit different for other founts.
`noadaptivecolumnsep`	Default: gives you the standard `\columnsep`, which can be changed by any of the `colsep` options above

The `\columnsep` produced by the `adaptivecolumnsep` option can be scaled by any of the `colsep` options.

2.5.7 Paper size

`rmpage` knows about three types of paper size options: main size, long size, and orientation.

Orientation—landscape or portrait

There are two options to select the paper orientation:

`landscape` Ensures that the longest side is horizontal
`portrait` Default. Ensures that the shortest side is horizontal

Make sure you only use one of these at a time.

Long sizes

A long paper size is based on a larger paper size; it formed by cutting off the specified fraction of a parent paper size, divided along the longer edge. For example, the common long size 2/3 A4 is 210 mm × 2/3297 mm = 210 mm × 198 mm.

These sizes are only formally defined for ISO A and B sizes. `rmpage` will make any main paper size into a long size with one of these options:

notlongpaper Default. Does nothing.
7/8longpaper Multiply the parent paper size length by 7/8
3/4longpaper Multiply the parent paper size length by 3/4
2/3longpaper Multiply the parent paper size length by 2/3
5/8longpaper Multiply the parent paper size length by 5/8
1/2longpaper Multiply the parent paper size length by 1/2
3/8longpaper Multiply the parent paper size length by 3/8
1/3longpaper Multiply the parent paper size length by 1/3
1/4longpaper Multiply the parent paper size length by 1/4
1/8longpaper Multiply the parent paper size length by 1/8

Make sure you only use one of these options at a time. They must be used with a main paper size option; the `letterpaper` main paper size option is used by default.

The resulting paper size is made `portrait` by default, or `landscape` if you've used that option, and printing limits are calculated based on the assumption that you will be printing on the parent paper size. That is, `rmpage` assumes that if you've asked for 2/3 long A4, you'll be printing on the top 2/3 of a sheet of A4, not a cut sheet of 2/3 A4. Or that if you've asked for 1/4 long A4, you'll be printing 4 pages on one sheet of A4.

See the chapters 6 and 7 to find out how and why this is done, and how to change these assumptions.

Main sizes

There's a lot more paper sizes available now; have a look at sections 5.1.2 and 5.2.1 for the full list. Any of the main paper sizes can be turned into a long paper size (see the section on long paper sizes above), and any paper size can be made landscape or portrait.

The available paper sizes include:

`letterpaper`, `executivepaper`, and `legalpaper`.	US paper sizes
`a0paper` to `a10paper`	ISO stationery sizes
`b0paper` to `b10paper`	ISO poster sizes
`c0paper` to `c7paper`	ISO envelopes
`dlpaper` and `c7/6paper`	ISO envelopes
`no10envelopepaper`	US envelopes
`foolscapefoliopaper`	obsolete stationery

There's over 70 sizes in all—to find out how paper sizes are declared to `rmpage`, and how to add new one, see chapters 6 and 7.

2.5.8 Founts

The standard LaTeX classes calculate a `\textwidth` on the assumption that you will be using Computer Modern Roman as the main body text fount. But the legibility of a line of text depends in part on the the width of a line measured in characters, and different founts have a different average character widths, so it's sensible to calculate a different `\textwidth` if you're using a different main body text fount.

`rmpage` will calculate an appropriate `\textwidth` if you use one of the options below to tell it what you are using as your main body text fount.

`avantwidth`	PSFNSS Adobe Avant Garde.
`bookmanwidth`	PSFNSS Adobe Bookman.
`chancerywidth`	PSFNSS Adobe Zapf Chancery.
`cmrwidth`	Default. Computer Modern Roman.
`concretewidth`	Donald Knuth's Concrete Roman.
`courierwidth`	PSFNSS Adobe Courier.
`helvetwidth`	PSFNSS Adobe Helvetica.
`lucasualwidth`	`bh` Lucida casual.
`newcentwidth`	PSFNSS Adobe New Century Schoolbook.
`palatinowidth`	PSFNSS Adobe Palatino.
`timeswidth`	PSFNSS Adobe Times.
`utopiawidth`	PSFNSS Adobe Utopia.
`thisfountwidth`	Bases `\textwidth` on the currently selected fount.

The `thisfountwidth` is useful if you are using a fount not covered by the standard options: it works by measuring the average character width of the fount that was selected when `rmpage` was loaded. For this to work properly, you must ensure that the main body text fount has been selected before loading `rmpage`. For example, if you are loading Adobe Baskerville in your preamble, you could ask `rmpage` to set an appropriate `\textwidth` like this:

```
\documentclass[thisfountwidth]{article}
\renewcommand{\rmdefault}{pgm}
\rmfamily
\usepackage{rmpage}
\begin{document}
...
```

If you haven't told `rmpage` to shut up with the `yorkshire` option, it will tell you which fount it's using as the basis for `\textwidth`—this is useful for people like me who get horribly confused by the details of fount selection.

If you want to use one of the PSNFSS packages to load a fount as well as set a `\textwidth` based on this fount, you can use one of these options:

`loadavant`	Requires the avant package.
`loadbookman`	Requires the bookman package.
`loadchancery`	Requires the chancery package.
`loadhelvet`	Requires the helvet package.
`loadnewcent`	Requires the newcent package.
`loadpalatino`	Requires the palatino package.
`loadtimes`	Requires the times package.
`loadutopia`	Requires the utopia package.

Each of these `loadfount` options does three things:

1. Loads the named package
2. Calculates a `\textwidth` based on the named fount
3. Sets the typesetting parameters to looser values.

The typesetting parameters are only loosened a little. The change does not affect the LaTeX commands `\fussy` and `\sloppy`, so using the `\onecolumn` and `\twocolumn` commands will over-ride this change. You can duplicate the effect of this loosening with the `\sloppiness` command—see section 2.5.12 on page 27 for more details.

If you've asked for a `twocolumn` layout, you get typesetting parameters close to the standard LaTeX sloppy values, unless you over-ride this looseness. Because the multicol package makes its own arrangements, you don't get the sloppy values if you asked for `twocolumnwidth` to `tencolumnwidth`; `rmpage` sets the typesetting parameters as if you were using a one column layout. Please email me if you have any thoughts on this matter.

The three `loadfount` options below are a little different to the PSNFSS fount loading options above:

`loadconcrete`	Requires the beton package; calculates `\textheight` based on beton's modified `\baselineskip`; sets `\textwidth` for Concrete Roman; doesn't loosen typesetting. See section 2.5.11 on page 25 for more on beton.
`loadcourier`	Makes the default roman fount Courier, sets an appropriate `\textwidth`, and asks for `loose` typesetting. I think this is ugly and crude: you might be better off using the times package and `\ttfamily`
`loadlucasual`	Requires the lucasual package; sets loose typesetting and a `\textwidth` to match Lucida Casual. Needs the lucasual files; available from CTAN at `fonts/psfonts/bh/lucasual/`.

2.5.9 Printers

It can be useful to let `rmpage` know about your intended output device, because it can ensure that the layout it produces will fit inside the printable region of the paper. At the moment, there aren't very many printer options that match real printers. As I get more information, I shall add more real printer options.

You can add an option for your own printer, or change the way your `rmpage` installation set non-printing margins for your printer—the details are described in section 7.4 on page 81.

The available printer options that I'll admit to here are:

`fullbleedprinter`	Prints right to the edge of the paper
`generalprinter`	This should be fine for anyone
`optimisticprinter`	
`pessimisticprinter`	Uses the largest non-printing margins I've found
`dw500printer`	Any HP 500 series inkjet
`dw600printer`	Any HP 600 series inkjet

2.5.10 Date format

You can change the way the `\today` command prints the date with the options below:

`ukdate nicedate`	5th November 1693
`usdate othernicedate`	Default: July 4, 1776

Only the `ukdate` and `nicedate` options make any changes: the `usdate` and `othernicedate` options do nothing.

The options names happened like this: I once wrote a LaTeX style file called `nicedate`, which produced the same effect as `rmpage`'s `nicedate` option—I like dates printed like that, you see. When I included the `nicedate` code in `rmpage`, it made sense to add a complementary option; hence `othernicedate`.

`othernicedate` seems preferable to `nastydate`, but it's not terribly memorable, so I created the synonyms `ukdate` and `usdate`. I'm not keen on these option names, but I can at least remember them. If you can think of something different, please let me know.

2.5.11 The `beton` package

The easy way of using Frank Jensen's `beton` package—to use Donald Knuth's Concrete Roman founts—with `rmpage` is to pass the `loadconcrete` option to `rmpage`. This will load the `beton` package, and set vertical and horizontal layout parameters for the Concrete Roman founts:

```
\documentclass[loadconcrete,concrete-math]{article}
\usepackage{rmpage}
```

The example above tells `rmpage` to load the concrete founts using the `beton` package. All global options are passed to all packages, so `beton` and `rmpage` are passed `loadconcrete` and `concrete-math`. `rmpage` ignores `concrete-math`

and acts on `loadconcrete`; while `beton` ignores `loadconcrete` and acts on `concrete-math`.

Because the `beton` package changes `\baselineskip`, but the changes don't take effect until the `begin{document}` command has been executed, and `rmpage` needs to know about the value of `\baselineskip` when it's setting `\textheight`, the `beton` package needs special support in `rmpage`. Everything's taken care of if you use the `loadconcrete` option. If you want to load `beton` with a `\usepackage` command, you should do this:

- Load `beton` before `rmpage`
- Pass the `beton` option to `rmpage`
- Pass the `concretewidth` option to `rmpage`—see section 2.5.8 on page 23.

Like this, for example:

```
\documentclass[beton,concretewidth]{report}
\usepackage{beton}
\usepackage{rmpage}
\begin{document}
...
```

There's no need to pass the `beton` option to `rmpage` if you're also passing the `stdbaselineskip` option to `beton`, but it will do no harm.

`rmpage` needs code that is in `beton` v1.3, 5th March 1995, to get things right. This version of `beton` was current in August 1996. I've made `rmpage` check the definition of the `beton` command it uses, but if you have any doubts that `rmpage` is doing its job properly, you can try this:

```
\documentclass[beton,chatty]{article}
\usepackage{beton}
\usepackage{rmpage}
\begin{document}
\typeout{\the\baselineskip\space according to beton}
\end{document}
```

Look through the console output for the lines that look like this (the numbers will vary depending on paper size etc):

```
\textheight is:
   48 x 13.0pt + 10.0pt = 634.0pt
```

This is a report of the number of lines in the text body ($48 + 1 = 49$ in this case), and how the final `\textheight` is arrived at. The second number—13 pt in this case—is `\baselineskip` as seen by `rmpage`. If this number is the same as the `\baselineskip` according to `beton`—as reported on the console—everything's probably okay. (The last term in the sum above is `\topskip`).

2.5.12 Typesetting parameters

LaTeX's `\sloppy` command tells TeX to be less fussy about linebreaking. It's sometimes useful to be able to tell TeX to be less fussy than normal, but not as sloppy as `\sloppy`. A good example is typesetting with founts installed by Alan Jeffries's fontInst package—-for example, the founts in the PSNFSS bundle of packages. Because these founts have a tighter inter-word space to Computer Modern Roman, Alan Jeffries recommends slightly looser typesetting parameters to usual, but not as loose as `\sloppy`—the values were reported by Sebastian Rahtz in his 'Notes on setup of PostScript fonts for LaTeX2', 14th August 1994. Oh dear: but Rahtz reports in '`PSNFSS2e.tex`' (05/11/96) that fontinst now produces founts with looser inter-word spacing, and suggests that the extra looseness is no longer required. I reckon a little extra looseness helps (the founts do seem to be tighter than the computer modern family), so the `loadPSfount` options have been changed to use the `looseish` settings normally, or the `sloppyish` settings if you've asked for two columns.

(What fontInst mainly does is make `vf`, `tfm`, `fd`, and `sty` files from `afm` files, so you can use with LaTeX founts which haven't been created with Metafont. It's available from CTAN.)

`rmpage` has an option that selects similar typesetting parameters, and variants: two fussier and two sloppier. Each option can in effect be selected at any point in your document, using the `\sloppiness` command as shown.

`tight`	Default. Standard LaTeX `\fussy` settings. `\sloppiness{0}`
`looseish`	`\sloppiness{1}`
`loose`	Similar to Jeffries's suggestion `\sloppiness{2}`
`looser`	`\sloppiness{3}`
`loosest`	`\sloppiness{4}`
`sloppyish`	For two columns `\sloppiness{5}`

The default changes to `loose` if you have use a `loadfount` option which loads a FontInst fount. See section 2.5.8 on page 23 for more information.

Note that is *not* equivalent to ; because standard LaTeX's `\sloppy` and `\fussy` commands change fewer parameters to `rmpage`'s `\sloppiness` commands, the `\fussy` command won't give you fussy typsetting if used after one of `rmpage`'s typesetting commands or options.

Chapter 3

Old Using rmpage

3.1 Some options

rmpage is more useful when you start using options: there's quite a lot of them, grouped in fairly consistently named sets. For example, you can select rmpage's normal \textwidth by saying normalwidth. If you want a slightly larger \textwidth, use the widish option instead. The wide, wider, and widest options will give you a progressively wider \textwidth. On the other hand, you can use the narrowish option to get a slightly narrower \textwidth to standard, and the narrow,narrower, and narrowest options do exactly what you might expect.

When you are dealing with a set of options like the width options above (all 13 of them; there's four odd ones I've not mentioned), be sure you only use one at a time. rmpage doesn't check, and you can get unexpected results if there's more than one option used from each set.

Text width narrowest, narrower, narrow, narrowish, normalwidth, wideish, wide, wider, and widest select a progressively larger \textwidth. fullwidth gives you the widest \textwidth that'll fit inside the printing region; stdwidth selects the same width as you would get with the standard classes; oneinchmargins selects a \textwidth that gives you an average margin size of one inch—the inside and outside margins add up to two inches; halfinchmargins is similar, but the inside and outside margins add up to one inch rather than two.

rmpage looks out for the onecolumn and twocolumn options, and calculates a possibly larger \textwidth if you say twocolumn. The class file is responsible for telling TEX to set text in two columns, so these options shouldn't be passed to rmpage only.

If you're using the multicol package, you can pass to rmpage the options: twocolumnwidth, threecolumnwidth, fourcolumnwidth, and so on up to tencolumnwidth. This will give you a \textwidth based on that number of columns.

Text height shortest, shorter, short, shortish, normallength, longish, long, longer, longest select a progressively longer \textheight. fulllength gives you the longest \textheight that will fit inside the printing region; stdlength gives you the \textheight you'd get with the standard class.

Headers and footers The `headers` and `footers` option leave space for headers and footers respectively; `noheaders` and `nofooters` give you page layout designed for no headers or no footers, respectively.

`leastheadsep`, `lessheadsep`, `lessishheadsep`, `normalheadsep`, `moreishheadsep`, `moreheadsep`, and `mostheadsep` enlarge and shrink the gap between the top of the text body and the bottom of the header; they work by scaling the standard value—`normalheadsep` does nothing.

`leastfootskip`, `lessfootskip`, `lessishfootskip`, `normalfootskip`, `moreishfootskip`, `morefootskip`, and `mostfootskip` enlarge and shrink the gap between the bottom of the text body and the bottom of the footer; they work by scaling the standard value—`normalfootskip` does nothing.

Positioning the text body horizontally `rmpage` looks out for the standard LaTeX class options: `twoside` and `oneside`. `rmpage` will produce a layout either giving you a text body intended for printing on one side of the paper, or alternating with odd pages on a right-hand page, and even pages on a left-hand page.

The `centre` (or `center`) option places the text body on the page with equal margins to the left and right. The `notcentre` (or `notcenter`) option places the text body with possibly unequal margins to the left and right.

If you've asked for `notcentre`, you will find that the larger margin is on the inside (intended for ring-binding). This effect is produced with the `notstdmargins` option. The `stdmargins` options reverses this, so the larger margin is on the outside (just like the standard classes).

`leastoffset`, `lessoffset`, `lessishoffset`, `normaloffset`, `moreishoffset`, `moreoffset`, and `mostoffset` enlarge and shrink the difference between the larger and smaller margin. `leastoffset` gives you `centre`d printing; there are subtle differences between `leastoffset` and `centre`.

Positioning the text body vertically `lowest`, `lower`, `low`, `lowish`, `normalaltitude`, `highish`, `high`, `higher`, and `highest`, shift the text body up and down the page—they are referred to as the altitude option set in this document (I know, but do you have any better ideas?)

Paper and printers `landscape` and `portrait` force that orientation, whatever the size paper.

There's lots of new paper sizes: see section 3.4 on page 30. You can ask for `7/8longpaper`, `3/4longpaper`, `5/8longpaper`, and so on down to `1/8longpaper`. This calculates what's called a long paper size, based on the main paper size selected, by dividing the main paper size into the specified fraction along the long edge. For example, A4 is 210 mm × 297 mm; 1/3 long A4 is 210 mm × 99 mm.

`lj4printer` tells `rmpage` you're using a Hewlett-Packard LaserJet 4 printer, and it'll calculate your page layout using appropriate non-printing margins. There are similar options for several other printers; you can add your own printer if it's not already here, and set any to be your default. For more details, see section 5.2.10 on page 66.

3.2 Some other options

Several option sets have corresponding `touch` options. These options increase or decrease whatever the parameter is by an amount in between the current main option and the next one. Generally, the sequence of values is a smooth geometrical one.

For example, passing `wide, touchwider` to `rmpage` gives you a `\textwidth` a third of the way up from `wide` to `wider`. `wider, touchnarrower` gives you a `\textwidth` a third of the way down from `wider` to `wide`.

A touch option can be used with a main option (any option which doesn't have `touch` or `t@uch` in the name is a main option; not all main options have `touch` options). The `t@uch` options should only be used in class and package files. They have exactly the same effect as a `touch` option, so saying `wider, touchnarrrower, t@uchnarrower` gives you the same width as `wider, touchwider`.

Text width setting `paperwidthset` makes `rmpage` set the `\textwidth` going by paper-based `\textwidth` only; `characterwidthset` goes by character-based `\textwith` only, but still pays attention to the printing limits specified by the paper size and printer selected. `bothwidthset` is the default setting.

`paperwidthset` was created so I could produce a layout for a timetable, fitting on A4 landscape paper. Paying attention to the number of characters in a line is inappropriate for that job, hence the option. The other two options are the natural complements.

The `touchlonger`, `touchshorter`, `touchwider`, and `touchnarrower` options increase or decrease `\textheight` or `\textwidth` one third of the way (in a geometrical sequence) towards the next main increment. That is, if you've said `wide` and `touchwider`, the width you get will be one third of the way towards the width given by `wider`.

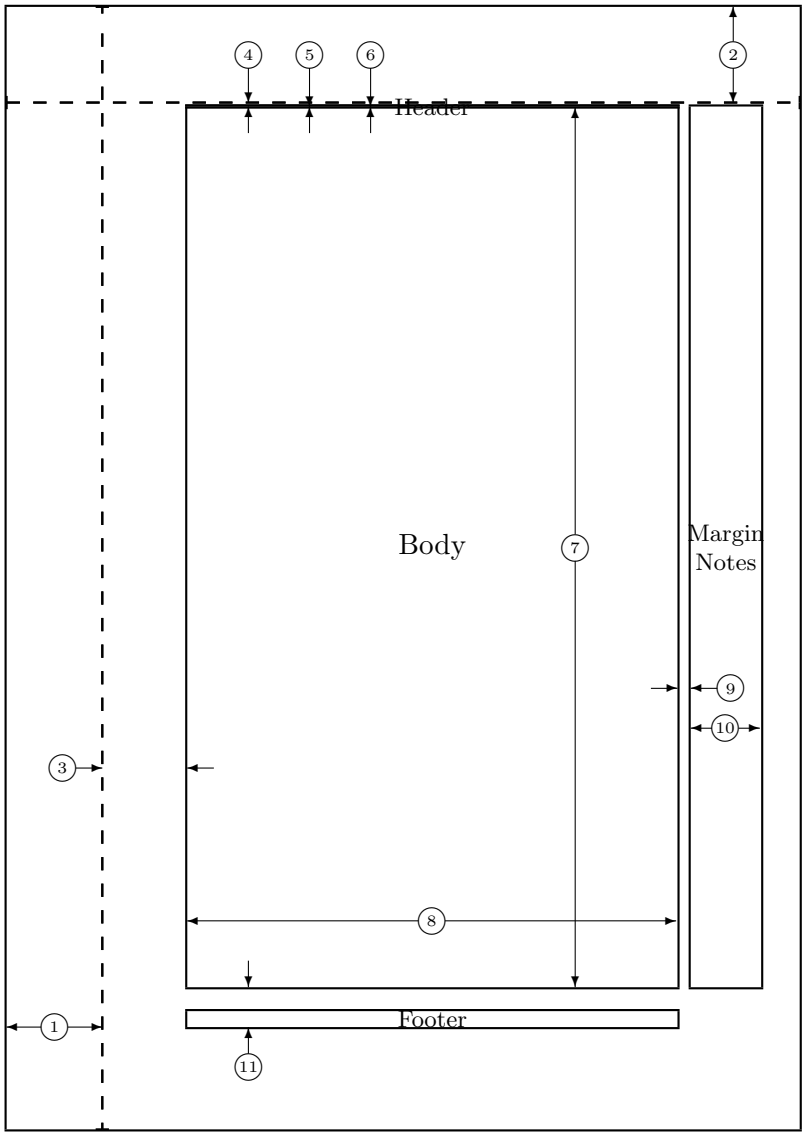
3.3 Layout details

The standard LaTeX package, layout, displays all the page layout parameters and their values. This document loads the package and uses the `\layout` command to display the values. You can see the results of the `\layout` command in figure 3.1.

Note that the layout for this document was calculated by `rmpage`, which was told to leave no space for a header with the `noheaders` option; that is why both `\headheight` and `\headsep` have zero size.

3.4 Paper sizes

`rmpage` knows about lots of paper sizes, including ISO long sizes. The ISO standard which defines the A and B series of sizes also defines long sizes. For example, A4 paper is defined as 210×297 mm. The long size 1/3 A4 is 210×99 mm—the long size is the base size divided into the specifed number of parts along the long edge. Because you can apply this division to any piece




```
 1   one inch + \hoffset          2   one inch + \voffset
 3   \oddsidemargin = 64pt        4   \topmargin = 3pt
 5   \headheight = 0pt            6   \headsep = 0pt
 7   \textheight = 663pt          8   \textwidth = 369pt
 9   \marginparsep = 10pt        10   \marginparwidth = 53pt
11   \footskip = 30pt                 \marginparpush = 5pt (not shown)
     \hoffset = 0pt                   \voffset = 0pt
     \paperwidth = 597pt              \paperheight = 845pt
```

Figure 3.1: The output of the `\layout` command

of paper you like, `rmpage` is happy to make non-ISO sizes into ISO-style long sizes. This means that Americans who want to print small booklets with two pages fitting on one sheet of letter paper can produce pages very easily with `rmpage`, by asking for `letterpaper` and `1/2longpaper`. You can do the same

thing with A4 paper, but why not just ask for `a5paper`? One commonly-used long size is 2/3 A4, which is often used for company invoices and commercial letters; it's often used in landscape orientation (210 mm × 198 mm) so it fits neatly into a world designed for A4-wide paper.

> If you are going to print two pages on one sheet of US letter paper, as I suggested, `rmpage` standard way of deciding what to set the non-printing margins to might be inappropriate. If so, look at the printer paper setting code in the configuration file, and specify clearances for the appropriate printer and paper combinations. Section 7.5 on page 81 has more details on this.
>
> A good reason for not asking for `a4paper` and then `1/2longpaper`, is that if the configuration file has particular settings for A5 paper, it won't apply them to this long paper size, even though it's the same physical size. A good reason for doing so is that you might want that.

`rmpage` accepts these paper options (and more):

`a0paper` to `a10paper`,
`b0paper` to `b10paper`,
`c0paper` `c7paper`,`c7/6paper`,`dlpaper`, `no10envelopepaper`,
etc

Options not listed here include some non-ISO envelope sizes, old British book sizes and so on: the odder sizes are kept in the configuration file. See section 5.2.1 on page 56 for more details.

You can get a long size by passing one of these options along with a paper size option: `notlongpaper`, `7/8longpaper`, `3/4longpaper`, `2/3longpaper`, `5/8longpaper`, `1/2longpaper`, `3/8longpaper`, `1/3longpaper`, `1/4longpaper`, or `1/8longpaper`.

`rmpage` makes an honest attempt to work out how you'll be printing these long sizes out (see section 6.6), but its decision might apply limits that aren't what you want. If so, you'll have to add some code to the configuration file to over-ride its guess. Section 7.5 on page 81 has more on how to do this.

By the way, because A3 printers aren't unheard of, and because you can print (say) 1/3 long A2 on A3 paper, it makes perfect sense to include an option to define A2 paper. It seemed churlish not to go all the way up to A0. 2/3 A4 is apparently a size commonly used for printing business invoices and the like. If anyone really does use `rmpage` for producing A3 pages, please let me know—it's something I've been wondering about.

If anyone would like to let me know about more US paper sizes, I'd be happy to include them in future versions of `rmpage`.

3.5 Setting the body text size

There's more detail on how this works in sections 6.1 and 6.2.

The range of values available for `\textwidth` and `\textheight` is a compromise. I wanted to keep the number of options down, produce a wide spread of values, with a small minimum step size.

The way things have turned out, the step size from one length option to the next is, in general, different to the step size from one width option to the next. So you can't maintain a particular balance of top:outside and bottom:inside margins (or whatever) by moving up to the next width and length option. There's no easy way round this—I wrote `rmpage` to set `\textheight` and `\textwidth` independently, because that seemed most sensible at the time. To ensure a fixed aspect ratio would mean I would have to re-write it to include code to scale up and down through a range of aspect ratios, which might have to be related to the aspect ratio of the paper, and allow you to scale the overall size of the text body up and down. This is a big job. I'd like to be able to set the text size that way as well as the way `rmpage` does it at the moment, but it'll have to wait—I suspect I'll have to re-structure the entire package to be able to support both ways of doing things without getting completely mixed up.

I decided to set both the width and height of the text body by scaling up and down geometric series. The `\textheight` series was chosen by deciding that the `longest` length should fill an ordinary page of A4, assuming a 6 mm non-printing border. The `normallength` length was set to the standard value (this is only strictly true if you're using headers and footers), which defined the geometric series. The `shortest` length on A4 is about 163 mm, just over half the page. It turns out that the series produces a central minimum step (`normallength` plus `touchlonger`) that is about one line (13 pt, near enough), so I think it's appropriate.

The `\textwidth` paper-based width series was derived similarly; the final values give a spread based on a `widest` value that fills an A4 page to within about 7 mm of the edge, and down to just over half the page. The `\textwidth` character-based width series has been more of a headache; the current (`rmpwnorm.pko v 0.52`) version gives a spread from about 75 characters in the middle, to about 98 and 57 characters at the extremes. The central minimum step (`normalwidth` plus `touchlonger`) is just under 2 characters, which seems suitably small. The problem with the character-based width setting is that the standard `\textwidth` is at the upper limit for easy reading; the widest available width, being wider than this, is far, far, far too wide for easy reading; but the minimum `\textwidth` doesn't get close to the minimum `\textwidth` for easy reading. But that probably won't matter too much; the minimum width is less than half an A4 page if you're using 11 pt Computer Modern Roman in one column, which seems small enough for most things I can think of (you might want a layout which places figures and extensive side notes in the margin; if most of the marginal note space were to be filled, having a very large margin and small `\textwidth` makes sense.)

3.6 Founts-

3.7 Marginal paragraphs-

`rmpage` looks at the everything including the sunspot cycle and the phase of the moon when it calculates the width of marginal paragraphs. Section 5.2.2 on page 59 lists all the options that control marginal paragraph size, and has

some more information.

Note this: the default margin for marginal notes is the outside margin for twosided printing, and the right-hand margin for one sided printing. The standard LaTeX command `\reversemarginpar` will reverse the marginal note placement; `\normalmarginpar` will put it back to normal. If you are printing twocolumn, the marginal notes end up in the nearest margin.

`rmpage` pays attention to all of this, and makes the marginal notes as large as possible given the margin they will appear in. The size is constrained by various parameters, controlled by the `maxmparwidth`, `mparsep`, and `mparclearance` sets of options.

The `clearance` parameter—the length `\RM@mparclearance`—is the minimum gap between the marginal note and the edge of the paper, subject to the additional restraint of the available printing area. It's hardwired to 0.4 in in standard LaTeX 2_ε; `rmpage` sets it to be a fraction of `\paperwidth` which works out to be 0.4 in if you're using US letter paper. The `sep` parameter—the length `\marginparsep`—is the gap between the marginal note and the body text—this is a standard LaTeX parameter. The `maxwidth` parameter—the length `\RM@maxmparwidth`—is the maximum allowed width: 2 in in standard LaTeX, a fraction of `\paperwidth` which works out to 2 in if you're using US letter paper with `rmpage`. There are also the `small` and `large` `basemparclear` and `basemaxmparwidth` options. The large versions of these options set the normal parameter size to twice the usual default value.

So if you're going to switch to `\reversemarginpar` in your document, do so *before* you load `rmpage`. If you're going to switch from one to the other, before you load `rmpage` select the one which allows the least space for marginal paragraphs.

If you're deeply interested in the workings of marginal paragraph size and placement, have a look at the `dtx` files: I think the basic idea is obvious. Note that the parameters `\RM@mparclearance` `\RM@maxmparwidth` can be set to any positive value by one of the hooks (`\RM@PrinterPaperSettings` might be a good one to use) in your configuration file: negative values are used by way of flags to set initial values.

3.8 The configuration files

`rmpage` uses a configuration file to do lots of stuff. The way it works is this: if you're LaTeXing with `rmpage` a file called `ermintrude.tex`, and a file `ermintrude.rmp` exists on the TeX search path, then `ermintrude.rmp` is used by `rmpage` as the configuration file for that run. If not, then `rmpage` looks for a file given by the command `\rmplocal`. If that file exists (if the command hasn't been defined before `rmpage` is loaded, it's set to `rmplocal`; the file is searched for with and without the `cfg` extension added). If that file can't be found, `rmpage` looks for `rmpgen.cfg`, which is a standard configuration file with everything enabled.

The file `rmplocal.gfc` is meant to be renamed `rmplocal.cfg` and changed locally. Have a read of the comments in the file first, and change the optional argument to the `\ProvidesFile` command to say that the file's been changed by you.

The idea is that you don't change `rmpgen.cfg`; it exists so everyone has a standard configuration file that will produce identical results. You can tell `rmpage` to use it in any given document by saying:

```
|\newcommand{\RMconfigfile}{rmpgen.cfg}|
...
\usepackage{rmpage}
```

in the preamble of the document, *before* loading `rmpage`. But please do change `rmplocal.cfg`, if you like.

The configuration file exists for these things:

- Declare additional standard options
- Declare user options
- Set local defaults
- Place code in various hooks to be executed inside `rmpage`, for supporting local classes, particular combinations of printer and paper, and so on.

Every installation of `rmpage` should have a modified configuration file for at least one reason: option processing is slow, and the fewer options you have, the faster it works. Folk with twin 225MHz PPC604e processors, or something huge and humming with the word 'Sun' on the front probably don't care about LaTeXing speed, but I do.

The idea is this: edit `rmpgen.cfg` and comment out all the options you think you won't often use. You can always uncomment them later. There's quite a few of these standard options that are commented out to begin with; you might want to uncomment some of them (if, for example, you're in the habit of printing on crown folio paper). By the way, don't add or delete anything anywhere above the local option declaration section; if you do, upgrading to a new version of `rmpage` might turn out to be more awkward than you'd like, and I'll come round and pester you, steal all your milk and put my feet up on the sofa.

Setting local defaults isn't hard either: there's an `\ExecuteOptions` statement with my local defaults in it. Take them out and replace them with what you want. If you use a LaserJet 4 printer and US letter paper, and most of your output goes into ring binders, say:

```
\ExecuteOptions{lj4printer,letterpaper,usdate,notstdmargins}
```

Note that I've assumed someone printing on US letter paper want dates formatted US style. If you don't want `notstdmargins`, say `stdmargins`.

There are some options that you can't specify in any `\ExecuteOptions` statement seen by `rmpage`: the `touch` options aren't allowed—they must be executed after their corresponding 'ordinary' options, which can't be arranged if they're bunged in an `\ExecuteOptions` statement. Try it if you like: `rmpage` will whinge at you.

If you want to declare a new printer type, for example, just copy one of the existing option declarations to the section marked 'BEGIN LOCAL OPTION DECLARATION', change the name and the parameters, and there you go.

You might want to set up particular printer/paper settings; there's a place for that and an example of how to do it in `rmplocal.cfg`. Have a look and a fiddle: if I could do it, it shouldn't be hard for someone as clever as you.

Adding code to the hooks in a useful fashion is best left to those who like fiddling around: have a look at the guts of `rmpage.dtx` and so on. Hopefully you'll get the hang of how I did things, and you'll be able to add code to suit your installation. I intend to document the whole thing properly eventually, but I only have 24 hours in each day. Try to stick to the conventions I've established: it'll make it more likely that your code will work properly with future versions of `rmpage`. My intention is to keep things more-or-less as they are, but this is the first version on public release, and I don't know what changes will be needed yet.

3.9 Sending `rmpage`-formatted documents elsewhere

There's a few potential problems with sending to other people documents that you've formatted with the help of `rmpage`. One is that they might not have `rmpage`, and another is that even if they do have `rmpage`, it is possible that you have specified options that produce layout parameters that are dependent on your particular printer, which your recipient might not have, and even if they do have that particular printer, they might have local configuration code—particularly printer/paper specific settings—that produces a slightly different result.

There's several different things you might do, depending on the circumstances—the thing to do is think about where your document's going, what's going to happen to it, and how you want it formatted.

In the usual run of things, a document that's been formatted with LaTeX and intended for general release is likely to be printed on A4 paper and US letter paper, so you have installation-dependent differences[1] even without `rmpage`—it's always worth checking that this difference won't cause a problem (so US authors: check that your document works on A4 paper, and everyone else in the world make an effort for those isolated Americans. Look, just go metric, will you? I read maps in miles and drink beer in pints[2], but really, metric units make life so much easier. If us *and* the Italians have made the switch, so can you. Italians? Yes, Italians: just which empire do you think is referred to in the phrase 'Imperial units' Yes, that's right, the Roman Empire. How about advancing into the 19th century before the 21st begins, eh? Sorry, a minor rant, but one that I think should be made occasionally.)

Ignoring paper sizes for the moment, if your recipent has a copy of `rmpage`, format your document with the `pessimisticprinter` option, ensure that the result is acceptable, and send the thing.

If your local configuration file has non-printer-specific settings that affect the output, you might send a copy of it (included in the preamble of your file before the `\usepackage{rmpage}` command, using the `filecontents` environment) under the name `jobname.rmp`. That is, if the document is in a file

[1] A4 paper is 210 mm × 297 mm or 8.27 inches × 11.69 inches; US letter paper is 8.5 inches × 11 inches or 215.9 mm × 279.4 mm

[2] real pints, not your sawn-off US version

called `canes-venaciti.tex`, call the configuration file `canes-venaciti.rmp` and it will be used as the configuration file for that document only.

Or you could run LaTeX with the `chatty` option specified to `rmpage`, and copy all the page layout parameters from the console window and paste them into the preamble of your document. If your version of LaTeX doesn't allow you to copy text from the console window (OzTeX 2.0 for the Macintosh does; I don't know about other versions), copy the text from the `log` file. Then comment out the call to `rmpage`. The problem with this is that it fixes everything, including paper sizes, so a recipient who uses A4 paper won't be impressed if they[3] get a document hard-formatted for US letter paper, and vice-versa. Of course, you could provide two sets of page layout parameters, with a note to the recipient to choose one or the other. This is messy, but it might produce the most reliable and legible results.

A possible way round the paper problem is to assume that no-one will be printing on anything other than US letter or A4 paper, and the people receiving the document won't mind if the layout isn't very good. That way, you might use the `letter4paper` paper size. This paper size is 210 mm×8.5 inches, and documents formatted with it will fit on A4 and US letter paper without formatting changes. Mind you, the results won't be very nice, and worse on US letter paper than on A4—`letter4paper` pages printed on A4 will be a little too short, which enlarges the gap at the bottom. Very often, this isn't too noticeable. But `letter4paper` pages printed on US letter paper will have the text body too far to the left, which tends to look very awkward.

3.10 Speed and what to do about it

`rmpage` chews up a lot of processor time when it is being processed at the start of a LaTeX run, but adds nothing to subsequent processing time—all it does is change page layout parameters. Narrow columns or short pages give TeX a harder time in line and page breaking, which might increase processing time, but I wouldn't worry about that if I were you—the underfull `\hbox`es and `\vbox`es will cause you more problems that the extra 0.2s time.

One verion of `rmpage` (v0.65) added 18.5 s to a LaTeX run on my Mac. The same version of `rmpage` added 19.1s to the processing time when I used it with the comments left in (before being processed by docstrip). This is an increase of 0.6s or 3%. Version 0.86 added 16 s with comments, or 15 s without, an increase of 7%. In this case, file size was reduced from 200 K to 85 K. It seems that, on my computer, the main benefit of using docstrip to remove comments from an input file is reduced file size, not reduced processing time. A not-terribly-formal test concluded that each character in the first argument to each `\DeclareOption` command added about 10 ms to processing time—I saved 1.6 s with version 0.72 by commenting-out eight options containing 176 characters.

My computer is a Macintosh Performa 475 12/160 with a 25MHz 68LC040 microprocessor; I am told this processor is roughly equivalent to a 40MHz 80486SX. For the tests noted above I used OzTeX version 2.0.1 and 2.1 under system B1 7.1 P5 SU3.

[3] A useful way of avoiding he/she that has an ancient history

You can speed up `rmpage` quite a lot. A large amount of the time taken by `rmpage` is in processing options—LaTeX 2$_\varepsilon$'s option processing mechanism is very slow. Commenting out options in the config file is very effective at reducing processing time (it's the length of the option names, not the amount of code, that increases this processing time).

For example, `rmpage` version 0.66 increased the time to process a document by 21.2s; with 73 options commented out, `rmpage` only added 12.1s. This is an improvement of nearly 60%. This is why there are two config files supplied with `rmpage`: a slow one with all the options enabled, and a faster one with some options commented out.

You can make `rmpage` work faster by commenting out (don't delete them—you never know when you might need them) all the options you think you won't use very often. If you do want to use one of these commented-out options, uncomment it (and all the options in the same group—you'll see what this means when you look at the file) and leave it uncommented. The Alpha text editor for Macintoshes can comment out a group of lines if you select the lines and press `cmd-D`; it can reverse the process if you press `cmd-opt-D`. I expect other text editors can do the same job somehow: it might be worth finding out how to do this with your text editor if you don't already know

Some versions of TeX under some operating systems take quite a time to find files to be input. If this is the case on your computer, you might be able to reduce processing time quite a lot by placing frequently-used files in a place that is searched early on. I have a list of folders in my `TeX-inputs` folder that looks a bit like this:

```
aa LaTeX
ab Rat2e <--- rmpage is in here
ab tools
ac mfnfss
ac other 2e <--- other people's packages
ac other fd
ac PSNFSS
ba .cfg files etc
ba other 2.09
ba Rat old
bb chicken <--- Liverpool John Moore's University logo
bb new fds
bb other LaTeX
bb some AMS-LaTeX
za afm
za fontinst
za fontinst examples
za fontinst inputs
zz Graphics
zz Plain
zz Rat ex + tmp
```

It's a compromise between speed and ease of management: I can change parts of my LaTeX system without getting a headache, and it's not too slow

for me. The commonly-used folders are prefixed `aa`, `ab`, end so on; the rarely used ones are prefixed `zz`. My version of LaTeX is set up to search the folders early on in the alphabet first: this is not necessarily the case with any other version of LaTeX, even versions of OzTeX with the `tex-inputs` list specified differently.

Some TeX implementations take a lot of time to search multiple directories, and are faster with fewer directories to search. If this is the case with your implementation, putting your input files into fewer directories might help. The only way I know to find if this is the case is to sit down and do lots of tests: it might be quicker in the long run not to bother.

Chapter 4

A brief lecture on typography

I'm not a professional typographer, and this won't turn you into one. Caveat emptor[1], and remember how much this cost you. If you want to learn about typography, do the sensible thing and get some books out of the library. But given that I'm giving you typographical controls, I'd better give you some idea how to use them intelligently. If you have any expertise in typography and think you can do better than this, please do! You've got my email address...

The notes below are what I think are the main points of ordinary typography the average LaTeX user needs to know about, dealing with the areas of typography that `rmpage` gives you new control over; this had no pretensions to being a course in typography—visit that library!

4.1 Introduction

The usual reason for writing something is so that people can read and digest the content—the reader is usually not interested in the form of written work, just the words themselves. It follows that the form in which your words are presented should usually be un-noticed by the reader: easy to read, conventional, and pleasing to the eye. The eye-catching typography used by advertisers does have its place, but this short piece concentrates on mundane typography: setting chunks of text, one word after another, line after line, to form a document which is unobtrusive to read. After all, if you were battling with the mathematics of elastohydrodynamics, you wouldn't appreciate a layout that was as hard to decipher as the content.

First thing is this: it's easy to make a right mess of things when you can control the layout of a page, but many of the mistakes one can make are in setting the paragraph indentation, vertical space around displayed material, headings, and the like. `rmpage` doesn't touch these, so you can forget about fouling up that area of design for now. As for the things you can control: don't stray too far from the normal settings, and the results should be fine.

The easiest mistake to make is in setting the width of the body text—`\textwidth` in LaTeX terms. Typography texts warn that the most common fault is making the body text too narrow (less than about 45 average characters); but my experience is that the average non-typographer is more likely to make the body text too wide (more than about 75 average characters).

[1] buyer beware

The best advice to begin with is follow convention—most books on typography make the point that your design has succeeded if no-one notices it, so stick with the conventional. Of course, you've got to find out what conventional is. Look at professionally designed typography: at how the text body sits on the page of a book, and in magazines and newspapers—is the text high, low, or in the middle? closer to the binding edge or the outside edge? How big, in visual relationship to each other and the text body, are the margins top, bottom, inside and outside? What about columns of text in magazines and newspapers—how wide are they, what's the gap between columns like in relationship to the column width, the size of the average word space, and the page margins? How does the column width affect readability—think about the effect of column width on line-breaking and the flow of reading.

You'll notice that conventionally, lines of text longer than about 75 characters are avoided, as are lines less than about 45 characters; and the outside margin is usually larger than the inside margin, for the very straighforward reason that the outside margin is the one that's most delicate and most handled. If it's big, damage to the paper is less likely to deface the text. The bottom margin is usually bigger than the top margin for the same reason—and even though this reason is a sound practical one, if you produce a page with a smaller bottom margin than top margin, it looks very odd to most people. It's also fairly conventional for the aspect ratio of the text to match the aspect ratio of the paper—if the text is half the width of the paper, it'll be half the height as well.

The standard LaTeX classes produce a page layout that's generally in line with those conventions, so if you don't pass extreme options to `rmpage`, you can't go far wrong. Of course, if you're aiming for a particular effect for a good reason, such as filling a page with a time-table, you might want the longest and widest text area possible. If you find that you *are* using an extreme option (`longest`, for example), think very carefully about why you are doing it, what effect this option produces, and whether this is desirable. If you're just trying to cram that much text on one page, two columns is probably better.

Because typography is to some extent an artistic endeavour, there is no such thing as an ideal layout, although some are clearly inappropriate. But after a few hours at the keyboard, it's easy to lose the ability to judge the passing of time, let alone the effect a design has on a new reader. I find that it often pays to trust initial reactions, especially when comparing a set of alternatives. So, if you're in doubt about which of your final two choices to use, ask someone else what they think of them.

4.2 Positioning the text body

Convention has it that the outside margin is larger than the inside margin, and that the bottom margin is larger than the top margin. It's not unusual to match the sizes of the outside margin and the top margin. The Koma-script documentation reports that Jan Tschichold recommends that the bottom margin should be twice the size of the top margin, and the outside margin should be twice the size of the inside margin.

The reason `rmpage` defaults to having the inside margin larger than the out-

side margin is this: most of what I typeset ends up in ring-binders, where you need a large inside margin to avoid punching holes in the text. You can change this default setting by editing an `\ExecuteOptions` statement in the configuration file—see chapter 7. You can change this default in the `rmplocal.cfg` file by putting the `stdmargins` option in the default `\ExecuteOptions` statement. If you don't like the amount by which the inside margin is bigger than the outside margin—bearing in mind that one convention has the outside margin matching the top margin—you can change it with the `offset` option set.

If the top margin looks bigger than the bottom margin, the page looks 'bottom heavy' and rather odd. It is possible that the only reason for this is that everyone lays out pages with a smaller top margin, but follow this convention to begin with. `rmpage` actually makes the top and bottom margins the same size, just as the standard classes do; the bottom margin looks larger because the bottom margin *appears* to begin at the bottom of the text body; LaTeX measures it from the bottom of the footer.

i couldn't figure out a reliable way of getting `rmpage` to calculate *apparent* ratios of top and bottom margins, so you'll have to balance this by eye. The `altitude`, `long`, and `short` option sets can help.

4.3 Size of the text body

Conventionally, one text column is 1.5 to 2.5 alphabets wide; the standard LaTeX widths give you about 2.5 alphabets wide (which is about the average width of 75 characters of normal prose, hence Lamport's comment in the LaTeX book. 1.5 alphabets is about 44 average characters, but `rmpage` warns when you get below 39 characters wide—TeX's a better line breaker than a human typesetter. `rmpage` warns you if you exceed the 75—39 character limits; too wide and too narrow are both awkward to read, and too narrow makes for rotten line breaks.[2] `rmpage` does report the final `\textwidth` in terms of what it thinks are average characters, which should help to give you an idea of what's going on.

If you must fill a very wide space, try using multiple columns. If you must use narrow columns, ragged right setting sometimes looks better, although a multiple-column narrow-columned layout might need to have `\columnsep` adjusted with `rmpage`'s `colsep` options—look at a newspaper to get the idea.

The problem with long lines is this: when you get to the end of one line, you need to find the start of the next one. If the lines are too long, this job is made harder. The problem with short lines is twofold: the job of finding the start of a line is a bother, so the less you have to do it the better; and short lines make for rotten line breaks, which makes it harder to follow the text.

Tschichold recommends that the aspect ratio of the text body be made to match the aspect ratio of the paper. This refers to the *apparent* aspect ratios—apparent paper size depends on the binding used, and apparent text body size depends on the nature of the headers and footers. I couldn't come up with a reliable way of calculating the comparative apparent aspect ratios,

[2] I'm lying: `rmpage` warns when you exceed the standard number of characters per line, which is 78.5 characters for 10pt, 74.8 for 11pt, and 75.5 characters for 12pt.

so if you want these aspect ratios matched, you'll have to do it by eye. The `altitude`, `offset`, `long`, `short`, `wide`, `narrow` option sets can help. Note that to my eye at least, matching aspect ratios often matters less than matching top and outside margins, which does appear often to make a big difference to my comfort with a layout on small (A5 or so) pages. And in any case, the match need not be spot on—but quite what near enough is can't be specified exactly. Sometimes an almost-but-not-quite match is fine, sometimes it looks awful and you're better off with a deliberate 'I'm clearly not trying to match these dimensions' layout.

4.4 Typefaces

The main thing the average LaTeX user is concerned with is legibility, and the computer modern typefaces score highly here. I have just seen a volume of conference proceedings (Proceedings of the Applied Optics Divisional Conference of The Insitute of Physics, held at Reading 16–19 September 1996) in which most papers were produced with LaTeX, using the publisher's style file. To my eye, the papers typeset in Times look most legible at a glance; the papers typeset in Computer Modern feel easier to read when you get down to it. I mention this to make the point that just because something looks right, doesn't mean that it's easier to read—not that Times is anything but a famously legible typeface.

As a general rule, serif typefaces are easier to read in blocks of text, and sans serif typefaces are easier to read as isolated words or phrases. This explains the choice of typeface on road signs and in newspapers. Typefaces with a heavy emphasis on vertical strokes—such as the Bodoni beloved of US newspaper headlines—disrupt the left-right flow of reading, whereas typefaces with a more horizontal emphasis, such as Times or Baskerville, ease the left-right flow of reading.

Choosing founts is a tricky thing—read those typography books! But there are some rules which can help you produce pleasant, legible pages. Before a real expert shoots me down in flames, I don't claim that the list below is 'rules for the correct use of typefaces'—it's just, erm, 'received wisdom', sort of.

- Always use a serif typeface for your body text.
- Keep the number of type faces and type families in a design to a minimum—use bold, italic, different sized, and maybe slanted type faces from the one family where appropriate.
- Always follow convention—don't invent a new convention unless you really need to.
- Don't mix similar but different typefaces—Times for the body copy and New Century Schoolbook for captions looks awful.
- But do mix very different typefaces—Times for the body copy and Helvetica (which I personally loath with a deep loathing) for captions or headings, for example, can be very effective.

- Use a body text size from 9 pt to 12 pt; some books suggest no more than 11 pt for body text, and I reckon 10 pt is a bit too small at 300 dpi, but fine at 600 dpi.
- Never, ever, underline unless you absolutely have to on pain of severe dandruff.
- And George Orwell said in his rules for good English, break any of these rules rather than do anything outright barbarous.

Chapter 5

All the options (rmpage v0.69 and rmplocal.cfg v0.11)

Most of the options in `rmpage` work by setting an internal paramters, which is later used to decide what value to set something to as part of a more involved calculation. Sometimes more than one parameter is used in this decision. The description of each option tells you what this parameter is set to, and what effect the option has.

5.1 Options in `rmpage`

This is a list of the options contained in the file `rmpage`; there's lots more in the configuration file. None of the options in `rmpage` are commented out, nor are any of the options in the configuration file `rmpgen.cfg`. Some of the options in the configuration file `rmplocal.gfc` have been commented out for speed's sake; this file will be used by `rmpage` if you rename it `rmplocal.cfg`

5.1.1 Reporting dimensions and tracing calculations

These options control the amount of stuff that `rmpage` litters your console window with. They do this by setting the `\RM@chatlevel` parameter, which is looked at by a bunch of reporting commands—look at `rmpage` for the details about these. The default is `taciturn`; you get a handful of dimensions and warnings get printed on the console. I expect that most people won't find `garrulous` useful. The `yorkshire` causes `rmpage` to print nought: this is a British regional joke that I'm allowed to make because I live in Lancashire, which is another British regional joke. (C.f. Ian MacMillan on '4th Column', BBC R4 6/10/96: "'Ey oop. All right. That should be enough for a column. Y'see, I'm from South Yorkshire, and we don't talk a lot.')

`garrulous` `rmpage` reports everything I thought might be useful someday. It used to be worse. Sets `\RM@chatlevel` to 0

`chatty` `rmpage` reports all LaTeX layout parameters that it changes, plenty of `rmpage`'s own parameters, and some information about what's going on as it caluculates them. Sets `\RM@chatlevel` to 1

`taciturn` Default. `rmpage` reports the height and width of the text and paper, the width of the text in characters, and a few other things. Warnings are also printed. Sets `\RM@chatlevel` to 2.

`yorkshire` Allows `rmpage` to print errors only, although warnings are put in the log file with a few other bits. Sets `\RM@chatlevel` to 3

5.1.2 Paper sizes

More paper types are defined in the configuration file, as are the long paper sizes. Each paper size is given a code number, because it's easier and faster to check for a number than a name. If you want to define your own paper sizes, I suggest you use code numbers above 1000 so that future versions of `rmpage` don't have standard sizes than conflict with your sizes. The paper size number 0 doesn't have much of a role in life yet, but I'm working on it.

The `landscape` and `portrait` options force the paper size to be in that orientation; the standard class swap `\textheight` and `\textwidth` when you ask for landscape. Landscape orientation is defined as the long side being horizontal; portrait orientation is defined as the short side horizontal. 2/3 A4 is usually used in landscape orientation (210×198 mm, as are DL envelopes. Some printer drivers, when dealing with DL envelopes, call 'landscape', 'portrait', and vice-versa.

`rmpage` knows about lots of sizes; some of them are large, obsolete, or untrimmed paper sizes which I don't expect will be directly useful to anyone. It just seemed inelegant to leave them out. Perhaps I have a warped sense of æsthetics. If you have a Macintosh with QuickDraw GX, you can tell your printer driver about any paper size that your printer is physically able to deal with, so these odd sizes might be more useful than I think.

If any Americans would like to send me the dimensions of some more US paper sizes (including envelopes), I'll include them in future versions. The only places I could find US paper size data were LaTeX classes and my printer's manual.

According to BS4000:

> The ISO A series is based on A0, with a surface area of $1\,\mathrm{m}^2$. Each ISO B paper size is a geometric mean between adjacent A sizes, with sides in the same proportions.
>
> Each size shall be acheived by dividing the size immediately above it into two equal parts, the division being parallel to the shorter side. Consequently, the areas of two successive sizes shall be in the ratio 2 : 1.
>
> All the size in each series shall be geometrically similar to one another.

What this means is that the ratio of the sides must be $1 : \sqrt{2}$, so that A0 is 841 mm × 1189 mm.

Tolerances are specified thus:

sizes $\leq$ 150mm	$\pm$1.5mm
150mm $<$ sizes $\leq$ 600mm	$\pm$2mm
600mm $<$ sizesmm	$\pm$3mm

Long ISO sizes are created by dividing ordinary ISO sizes into slices, cutting parallel to the short edge, e.g.,

1/3 A4 99×210
1/4 A4 74×210
1/8 A8 13×74

2/3 A4 is apparently a common size commercially, used for invoices and the like. It is defined as 198×210mm. Note that the standard defines sizes to the nearest millimetre, but `rmpage` does *not* round the calculated long sizes, nor does it ensure that only ISO sizes are processed by the 'long' options.

`rmpage` does not limit you to making long sizes out of ISO paper only because you can chop up any bit of paper you like. The reason `rmpage` does not round long sizes to the nearest millimetre is that if you are printing on a piece of ready-cut long paper, `rmpage`'s maximum deviation from the specified size, 0.5 mm, is within tolerance; and if you are making your own long paper by cutting up a straight size, rounding to the nearest millimetre on a 1/8 size could result in a 4 mm error by the time you cut the strip furthest from your datum edge, which is outside the specified tolerance for the size of the sheet you are cutting. The fact that not rounding is easier to code is, of course, entirely co-incidental and played no part in the design decision.

Data source for the old British book sizes: Pears Cyclopedia, 68th edition, 1959-1960. Pelham Books Ltd., 1959. (General Compendium, page N13). A, B, and other untrimmed sizes taken from BS4000. C3, C4, C5, C6, DL, and non-ISO envelope sizes taken from BS4264. C0, C1, C2, C7, and C7/6 taken from The Cambridge Factfinder, Cambridge University Press, 1993.

Each paper type is given a number:

0=undefined, 1=letter, 2=legal, 3=executive,
9=letter4paper
10=a0, ..., 20=a10 (a4=14, a5=15)
30=b0, ..., 40=b10
50=c0, ..., 57=c7, 58=dl, 59=c7/6,
60=bspopseedenvelope, 61=bspopnonisoenvelope,
62=bsbrochureenvelope, 63=bslegalenvelope,
64=bslargelegalenvelope, 65=bscalendarenvelope
66=no10envelopepaper
70=foolscap folio, 71=foolscap quarto, 72=foolscap octavo
73=crown folio, 74=crown quarto, 75=crown octavo
76=royal folio, 77=royal quarto, 78=royal octavo
79=imperial folio, 80=imperial quarto, 81=imperial octavo
82=large crown octavo
83=demy quarto, 84=demy octavo
85=medium quarto, 86=medium octavo
90=ra0, 91=ra1, 92=ra2,
93=sra0, 94=sra1, 95=sra2
96=metric double crown paper, 97=metric quad crown paper
98=metric large quad crown paper, 99=metric quad demy paper

100=metric small quad royal paper

The paper types are divided up like this:

3	US sizes recognised by 3 options	(1-3)
1	bodge size recognised by 1 option	(9)
11	A sizes defined by 11 options: for writing paper, books, etc.	(10-20)
11	B sizes defined by 11 options: for posters, etc.	(30-40)
10	C sizes plus dlpaper defined by 10 options: for envelopes, etc.	(50-59)
6	BS4264 envelope sizes defined by 6 options:	(60-65)
1	US envelope size defined by 1 option:	(66)
17	old British sizes defined by 18 options.	(70-86)
11	BS4000 untrimmed sizes defined by 11 options	(90-100)

71 different paper sizes defined by 72 options.

`letterpaper` Sets `\RM@papertype` to 1 – 11in by 8.5in
`legalpaper` Sets `\RM@papertype` to 2 – 14in by 8.5in
`executivepaper` Sets `\RM@papertype` to 3 – 10.5in by 7.25in
`a4paper` Sets `\RM@papertype` to 14 – 297mm by 210mm
`a5paper` Sets `\RM@papertype` to 15 – 210mm by 148mm
`b5paper` Sets `\RM@papertype` to 35 – 250mm by 176mm
`c6paper` Sets `\RM@papertype` to 56 – 162mm by 114mm
`dlpaper` Sets `\RM@papertype` to 58 220mm by 110mm. Ordinary envelopes.
`no10envelopepaper` Sets `\RM@papertype` to 66 9.5in by 4.12in. Ordinary US envelopes.

5.1.3 Typesetting tightness

These options only exists because `rmpage` can load the PSNFSS founts, and Karl Berry says that these founts (produced with FontInst) have too little slack in the inter-word space for TEX to be able to form paragraphs well with the standard typesetting parameters.

These options must be executed before the `load<fount>` options. That's inevitable if I have these option declarations before the `load<fount>` option declarations, and use `\ProcessOptions` rather than `\ProcessOptions*`. This is so that these options can over-ride the default typesetting tightness (`looseish`) requested by the `load<fount>` options.

`tight` Default. Leaves the typesetting parameters alone. Defines `\RM@looseoption` to be 0.
`looseish` Changes the typesetting parameters to be something in between the adjacent options. Defines `\RM@looseoption` to be 1
`loose` Changes the typesetting parameters to something close to Alan Jeffries's recommendations. Defines `\RM@looseoption` to be 2
`looser` Changes the typesetting parameters to be something in between the adjacent options. Defines `\RM@looseoption` to be 3
`loosest` Changes the typesetting parameters to something close to twice as sloppy as Karl Berry's recommendations. Defines `\RM@looseoption` to be 4

`sloppyish` Changes the typesetting parameters to be even looser than `\sloppy`, for two column typesetting with PSFNSS founts. Defines `\RM@looseoption` to be 5.

5.1.4 Textheight setting

These request a `\textheight` shorter or longer than normal. If you allow space for headers and footers with the `headers` and `footers` options, `normallength` gives you the same `\textheight` as you'd get with the standard classes. The other lengths are scaled up and down from the normal value in a geometrical sequence. (Actually, the total space above and below the text body including headers and footers is the dimension that's scaled in a geometrical sequence, but that shouldn't bother you too much.)

The `touchlength` options add or subtract one from the value set here.

There are also `stdlength` and `fulllength` options in the configuration file; see section 5.2.4 on page 62.

`shortest` Sets `\RM@lengthoption=3`; this number gives you: `\RM@totalheadfootclearance` = 0.5200`\paperheight`
`shorter` Sets `\RM@lengthoption=6`
`short` Sets `\RM@lengthoption=9`
`shortish` Sets `\RM@lengthoption=12`
`normallength` Default. Sets `\RM@lengthoption=15`; this number gives you: `\RM@totalheadfootclearance` = 0.2130`\paperheight`
`longish` Sets `\RM@lengthoption=18`
`long` Sets `\RM@lengthoption=21`
`longer` Sets `\RM@lengthoption=24`
`longest` Sets `\RM@lengthoption=27`; this number gives you: `\RM@totalheadfootclearance` = 0.0872`\paperheight`

5.1.5 Headers and footers

These options only deal with the gap between text body and the header (or footer). See section 5.1.12 on page 54 for options to turn the headers and footers on and off.

The way these options work is by setting a parameter which is passed to the `\RM@scalebyoption` command, to scale the required page layout parameter by a value in a geometric sequence. Have a look at the command in `rmpage` for the details.

`\headsep` is just multiplied by the requested value. `\footskip` is the distance from the bottom of the text body to the bottom of the footer. To approximate a scaling of the distance between the top of the footer and the bottom of the text, `rmpage` assumes that the footer is a single line that is `\baselineskip` high, and subtracts `\baselineskip` from `\footskip` before scaling, and adds it back afterwards. This seems to work well enough.

The value set by the options below can be modified by the `touchheadsep` and `touchfootskip` options, which add or subtract one from the appropriate parameter.

`leastheadsep` Sets `\RM@headsepoption=3`

`lessheadsep` Sets `\RM@headsepoption=6`
`lessishheadsep` Sets `\RM@headsepoption=9`
`normalheadsep` Default. Sets `\RM@headsepoption=12`
`moreishheadsep` Sets `\RM@headsepoption=15`
`moreheadsep` Sets `\RM@headsepoption=18`
`mostheadsep` Sets `\RM@headsepoption=21`
`leastfootskip` Sets `\RM@footskipoption=3`
`lessfootskip` Sets `\RM@footskipoption=6`
`lessishfootskip` Sets `\RM@footskipoption=9`
`normalfootskip` Default. Sets `\RM@footskipoption=12`
`moreishfootskip` Sets `\RM@footskipoption=15`
`morefootskip` Sets `\RM@footskipoption=18`
`mostfootskip` Sets `\RM@footskipoption=21`

5.1.6 Columnsep

`\columnsep` is a standard LaTeX parameter: it is the space in between columns of text on a multiple column page. The `colsep` options scale `\columnsep` using the internal `\RM@scalebyoption` command: see section 6.7 on page 77 for details. Briefly, `normalcolsep` does nothing; `mostcolsep` multiplies `\colsep` by 2.5; `leastcolsep` divides `\columnsep` by 2.5; and intermediate options use a factor in between along a geometrical sequence.

There are corresponding `touchcolsep` options. They must be executed after these main options, which is easily arranged—see section 5.2.3 on page 61 for the details.

`leastcolsep` Sets `\RM@columnsepoption` to 3
`lesscolsep` Sets `\RM@columnsepoption` to 6
`lessishcolsep` Sets `\RM@columnsepoption` to 9
`normalcolsep` Default. Sets `\RM@columnsepoption` to 12
`moreishcolsep` Sets `\RM@columnsepoption` to 15
`morecolsep` Sets `\RM@columnsepoption` to 18
`mostcolsep` Sets `\RM@columnsepoption` to 21

If `\RM@adaptivecolseptrue`, then `\columnsep` is set to be a fraction of the number of points per character. This isn't always appropriate, and the flag is set to false by default.

`adaptivecolsep` `\columnsep` is set to be 2.3 times the width of one average character, according to `rmpage`'s reckoning. This new value can be scaled by the `mostcolsep` to `leastcolsep` options.

The standard `\columnsep` is 2.03 times the width of one average 10 pt Computer Modern Roman character. This is a useful option if you are creating a style based on a fount size larger than 12 pt. Otherwise, it seems to be a good idea on Mondays, Wednesdays, and Fridays; not so good on Tuesdays, Thursdays, and Saturdays; and on Sundays, I write Ogham on tree bark.

`noadaptivecolsep` Default. `\columnsep` is not changed from its default value, although it might be scaled by the `mostcolsep` to `leastcolsep` options.

5.1.7 Width of the text body

The width options let you ask for a larger or smaller `\textwidth`. Following the basic idea of the standard classes, `rmpage` calculates two different `\textwidths`: one is based on the number of characters in a line; the other is based on the size of the paper. The smaller of these two guesses is used as the basis for the final `\textwidth`—`\textwidth` is also constrained by `\RM@mintextwidth`, `\RM@maxtextwidth`, `\RM@mininsidemargin`, `\RM@minoutsidemargin`, `\RM@minleftclearance`, and `\RM@minrightclearance`.

`rmpage` is inclined to print out warnings if it has to change its preferred `\textwidth` because of one of the above restrictions—the `yorkshire` option will silence these warnings if you find them irritating.

The `normalwidth` option gives a `\textwidth` close to the standard LaTeX width on US letter or A4 paper, where the character-based width is usually used (wide founts like Lucida Casual are the exception to this). `\textwidth` is larger than usual if you print on smaller paper, where the paper-based `\textwidth` is used. The options ranging out to `widest` and `narrowest` request a `\textwidth` varying in a smooth geometrical sequence, but remember that the smaller of the two widths (character-based and paper-based) is used, and there are several other restrictions on `\textwidth`, so this smooth progression may not be apparent as you step up or down through the options. The gory details are in the file `rmpnorm`.

You can control which of the two widths—character-based or paper-based—is used as the final `\textwidth`. See the next section (section 5.1.8) for details.

The `stdwidth` option forces `rmpage` to calculate the `\textwidth` in the same way as the standard classes, except that the final value is still subject to the restrictions listed above. So it is possible to ask for `stdwidth` and get a `\textwidth` that is not what you'd've got with the a standard class. `rmpage` will warn you if this happens.

`widest` Paper-based textwidth is set to 1.3096 times the `normalwidth` value; character-based textwidth is set to 1.9761 times the `normalwidth` value. Sets `\RM@widthoption` to 26

`wider` Sets `\RM@widthoption` to 23

`wide` Sets `\RM@widthoption` to 20

`wideish` Sets `\RM@widthoption` to 17

`normalwidth` Default. Paper-based `\textwidth` is set to `0.7138\paperwidth`; character-based `\textwidth` is set to 78.5 characters (10 pt), 74.8 characters (11 pt), or 75.5 characters (12 pt)—this produces a character-based `\textwidth` very close to the standard width setting code. Sets `\RM@widthoption` to 14

`narrowish` Sets `\RM@widthoption` to 11

`narrow` Sets `\RM@widthoption` to 8

`narrower` Sets `\RM@widthoption` to 5

`narrowest` Paper-based textwidth is set to 0.7636 times the `normalwidth` value; character-based textwidth is set to 0.5061 times the `normalwidth` value. Sets `\RM@widthoption` to 2

`stdwidth` Attempts to produce a page with the same `\textwidth` as the standard classes would give. Sets `\RM@widthoption` to 32

`halfinchmargins` Attempts to produce a page with a total horizontal margin space of one inch. If you have asked for `centre`d printing, `rmpage` will try to produce half inch margins either side. Sets `\RM@widthoption` to 31

`oneinchmargins` Attempts to produce a page with a total horizontal margin space of two inches. If you have asked for `centre`d printing, `rmpage` will try to produce one inch margins either side. Sets `\RM@widthoption` to 30

`fullwidth` Produces the widest possible `\textwidth` given all other restrictions. Sets `\RM@widthoption` to 29

5.1.8 Width setting control

These control which dimensions `rmpage` takes notice of when setting `\textwidth`—`rmpage` can look at the size of the paper and the number of characters when it's setting `\textwidth`. Normally it looks at both, and picks the one that results in the smallest `\textwidth`. The code that does this is in the width setting `pko` file; it works out what to do based on the value of the `\RM@setwidthby` command.

If you ask for one of these widths: `oneinchmargin`, `halfinchmargin`, and `fullwidth`, you shouldn't also ask for `characterwidthset`, because the width options ask for widths that are inherently based on the size of the paper. `rmpage` will point out this mistake if you make it, and carry on as if you'd not said `characterwidthset`.

I'm not that keen on these option names, especially `bothwidthset` which is formed in a regular sequence with the other two, and ends up both ugly and not very descriptive; if you can come up with something better, please let me know.

`bothwidthset` Default. Use both paper and character based `\textwidth` requests to set `\textwidth`. Defines`\RM@setwidthby` to be 0

`characterwidthset` Use the character based `\textwidth` request only to set `\textwidth`. Defines `\RM@setwidthby` to be 1

`paperwidthset` Use the paper based `\textwidth` request only to set `\textwidth`. Defines `\RM@setwidthby` to be 2

5.1.9 Margins

The options in this section control the horizontal position of the text body. The `twoside` and `oneside` options are standard options that `rmpage` understand; the rest of them are new in `rmpage`.

It's probably best to read about all of these options, not just some of them, because they all interact to some extent.

`twoside` Places the text body for printing on both sides of the paper, taking into account the requested offest and which margin you want to be the larger one (inside or out).

Sets `\RM@twosidetrue` and `\@mparswitchtrue`; the latter step is performed by the standard classes. I'm not certain this is the right thing to do; it means you get the standard classes' effect if you pass

this option to `rmpage` only. But you might want to avoid the standard classes' effect... But if you're that clever, you can reset the switch yourself.

`oneside` Default. Places the text body for printing on both sides of the paper, taking into account the requested offest and which margin you want to be the larger one (inside or out). Sets `\RM@twosidefalse`

`centre` Forces the left and right margins to be the same size. Sets `\RM@centretrue`.

`notcentre` Default. Allows the left and right margins to be different sizes. Sets `\RM@centrefalse`

The options in the list below control which margin is the larger one. Conventional book typesetting and LaTeX makes the outside margin the larger one; `rmpage` makes the inside margin the larger one. This is because most of what I produce is bound in loose-leaf ring-binders, where having a small inside margin often results in a holes in the text.

You can change this default setting by changing the `notstdmargins` option to `stdmargins` in the default `\ExecuteOptions` statement in your local configuration file. You will find this statement just below the line in the configuration file that reads: CHANGE THIS LINE TO MATCH YOUR LOCAL PREFERENCES.

`stdmargins` Default. Outside margin is the larger one. Sets `\RM@stdmarginstrue`

`notstdmargins` Inside margin in the larger one. Sets `\RM@stdmarginsfalse`

You can control the relative proportions of the inside and outside margins with the `offset` options. These offset options don't do anything if the centre option has been specified.

The default offset is 60% of the total horizontal margin space in the larger margin, 40% in the smaller. This the the standard LaTeX 2_ε offset. `leastoffset` gives you equal margins; `touchlessoffset` and `leastoffset` together makes the nominally larger margin into the smaller one, by a small amount. `rmpage` will warn you if this happens. `mostoffset` puts 87% of the total horizontal margin space into the larger margin; this is about as far over to one side as your printer is likely to be able to print.

There are `touchoffset` options in the standard configuration files—see section 5.2.3 on page 61

Have a look at `rmpnorm` for more details if you need them.

`leastoffset` 50% larger margin. Sets `\RM@offsetoption` to 2
`lessoffset` 53% larger margin. Sets `\RM@offsetoption` to 5
`lessishoffset` 56% larger margin. Sets `\RM@offsetoption` to 8
`normaloffset` Default. 60% larger margin. Sets `\RM@offsetoption` to 11
`moreishoffset` 68% larger margin. Sets `\RM@offsetoption` to 14
`moreoffset` 77% larger margin. Sets `\RM@offsetoption` to 17
`mostoffset` 87% larger margin. Sets `\RM@offsetoption` to 20

The touchoffset options must be executed after the offset options. This is easy to arrange: just declare the options with the `touch` options after the main options, and use `\ProcessOptions` rather than `\ProcessOptions*` (that is, these options must be processed in the order of declaration, rather than the order given in the calling commands).

5.1.10 Number of columns

Note that the standard classes set the `\@twocolumn` flag true or false, depending. `rmpage` doesn't, and works quite happily without it.

The config file has `onecolumnwidth` to `tencolumnwidth` options, which change `\textwidth` but don't change the number of columns that LaTeX typesets text in. You can use the multicol package to do that.

`onecolumn` Default. This standard option is recognised by `rmpage`. This option makes the standard classes typeset one column to a page; `rmpage` calculates a character-based `\textwidth` based on this. Defines `\RM@textcols` to be 1.

`twocolumn` This standard option is recognised by `rmpage`. This option makes the standard classes typeset two columns to a page; `rmpage` calculates a character-based `\textwidth` based on this. Defines `\RM@textcols` to be 2.

5.1.11 Paper orientation

These options really do force the appropriate orientation; the standard classes just swap `\textheight` and `\textwidth` when asked for `landscape`. Remember that you'll most likely want to print your envelopes landscape, even if your printer driver thinks you mean portrait (Hewlett Packard's DeskWriter series 6.0 printer driver gets this wrong. Oops.) And 2/3 A4 is usually used in landscape orientation, even though you'll probably think it's portrait—I know I did.

`portrait` Default. Forces `\textwidth` to be less than `\textheight`. Sets `\RM@portraittrue`

`landscape` Forces `\textwidth` to be more than `\textheight`. Sets `\RM@portraitfalse`

5.1.12 Headers and footers

Allows space for headers and footers, or not, as you wish. These options *do not* affect the contents of headers and footers in any way: if you want to change the LaTeX `\pagestyle`, you must do that separately.

Turning headers and footers on and off changes `\textheight`: this is because of the way `rmpage` calculates `\textheight`.

`rmpage` first calculates the sum of the blank space above and below all the text on the page; this is a constant fraction of `\paperheight` for any given length option. What is left over after space has been allowed for headers and footers is `\textheight`

`noheaders` Produce a layout for pages without headers. Sets `\RM@headersfalse`; this results in `\headheight` and `\headsep` being set to 0 pt.

`headers` Default. Produce a layout for pages with headers. Sets `\RM@headerstrue`; this results in `\headheight` being set to `\baselineskip`.

`nofooters` Produce a layout for pages without footers. Sets `\RM@footersfalse`; this results in `\footskip` being set to 0 pt

`footers` Default. Produce a layout for pages with footers. Sets `\RM@footerstrue`.

5.1.13 Positioning the text body vertically

These options affect the ratio between the gap below all the text and the gap above all the text. The `touchaltitude` options change this ratio in increments of 1/24.

`highest` Top:bottom space = 0:8. Sets `\RM@headfootbalance=0`
`higher` Top:bottom space = 1:8. Sets `\RM@headfootbalance=3`
`high` Top:bottom space = 2:8. Sets `\RM@headfootbalance=6`
`highish` Top:bottom space = 3:8. Sets `\RM@headfootbalance=9`
`normalaltitude` Default. Top:bottom space = 4:8. Sets `\RM@headfootbalance=12`
`lowish` Top:bottom space = 5:8. Sets `\RM@headfootbalance=15`
`low` Top:bottom space = 6:8. Sets `\RM@headfootbalance=18`
`lower` Top:bottom space = 7:8. Sets `\RM@headfootbalance=21`
`lowest` Top:bottom space = 8:8. Sets `\RM@headfootbalance=24`

5.1.14 Changing the date format

`usdate` Default. Sets `\RM@nicedatefalse`, which causes nothing to happen; the `\today` command is unmolested.

`ukdate` Sets `\RM@nicedatetrue`, which causes the `\today` command to be re-defined to produce a date of the form '4th April 1984'. This is the setting I use as a default; I do this by putting the `ukdate` option in the local settings `\ExecuteOptions` statement in my local configuration file.

5.1.15 Dealing with the beton package

The code to let `rmpage` work with `beton` felt rather complicated to write. The thing about the `beton` package is that it changes `\baselineskip` to something non-standard. `rmpage` needs to know what `\baselineskip` is so that it can set `\textheight`, but `beton`'s changes aren't made until the `\AtBeginDocument` hook is executed by LaTeX, which is after `rmpage` has been loaded. I had to steal code from `beton` v1.3 to deal with this, which might cause problems if you try to use `rmpage` with other versions of `beton`.

The result is that if you are using the `beton` package without passing it the `standard-baselineskips` option, you should specify either the `beton` or the `nobeton` option to `rmpage`: the first option uses `beton`'s modified `\baselineskip` to set `\textheight`; the second option uses the standard `\baselineskip`, and can be ommitted if you have specified the `standard-baselineskips` option to `beton`.

The problem with the `beton` option is that if you specify it, `rmpage` uses code stolen from the guts of `beton` version 1.3 to set the appropriate `\baselineskip`. There is no guarantee that this code will work with other versions of `beton`.

`beton` calculate a `\textheight` based on the `beton` package's `\baselineskip`
`nobeton` calculate a `\textheight` based on the standard `\baselineskip`

Both these options set the `\RM@ifbeton` command to a number. It's played about with before and after here. The final value of the `\RM@ifbeton` command is given these meanings within `rmpage`:

0 beton package loaded and the beton option specified
1 The beton package has been loaded with neither the beton nor the nobeton option specified
2 beton package loaded and the beton option not specified
3 The beton package not loaded with neither the beton nor the nobeton option specified
4 The beton package not loaded and the nobeton option specified.

5.2 From the configuration file

The following options are all from the configuration file. There's nothing magical about this: they could all just as easily be in rmpage.sty at the point where the configuration file is loaded. But the idea is that you can change the configuration file, but not rmpage, and rmpage works faster with fewer option. So comment out any of these options that you don't use very often (please don't delete them: you never know when you might need them).

The distributed configuration file rmpgen.cfg has no options commented out; this means it's quite slow. The distributed configuration file rmplocal.gfc does have some options commented out—rmpage works faster using this file. rmpage will not use rmplocal.gfc as a configuration file unless you tell it to. The most straightforward way to get rmpage to use this faster configuration file is to rename it rmplocal.cfg.

The idea is that you don't change rmpgen.cfg at all: it's intended to be a standard configuration file that any document can use to produce identical output on any system by saying \newcommand{\RMconfigfile}{rmpgen.cfg} in the preamble before the \usepackage{rmpage} command.

If you are short of disc space, you could delete rmpgen.cfg, but you might find you have to re-install it one day to process a file that requires it.

rmplocal.cfg is intended to be changed by anyone. Please read the comments in the file first, add a comment at the start of the file to identify it as yours, and a note to the same effect in the optional argument of the \ProvidesFile: life can get very confused otherwise. Don't add or delete anything except comment characters between the \ProvidesFile command and the line LOCAL CODE BELOW HERE PLEASE. Make sensible changes below the line LOCAL CODE BELOW HERE PLEASE; read the comments in the configuration file and rmpage.dtx, and use the commands I use for doing things. If you do this, your code should work perfectly with future versions of rmpage.

In the list below, options that look like this: *obscurefunction*, are commented out in rmplocal.gfc, whilst options that look like this: usefulfunction, are not commented out.

5.2.1 Other paper sizes

There are some notes on paper sizes in section 5.1.2 on page 46. I suspect that the larger sizes and untrimmed sizes will be useless, but it seemed churlish to leave them out.

undefinedpaper Sets \RM@papertype to 0 – does nothing to \paperheight or \paperwidth. This paper type has no purpose in life, yet.

letter4paper Sets \RM@papertype to 9; this paper size is an unholy bodge with the width of A4 and the height of US letter. Documents typeset with this paper size will fit on A4 and US letter paper, and look terrible on both. Size is 210mm by 8.5in.

a0paper Sets \RM@papertype to 10 – 1189mm by 841mm

a1paper Sets \RM@papertype to 11 – 841mm by 594mm

a2paper Sets \RM@papertype to 12 – 594mm by 420mm

a3paper Sets \RM@papertype to 13 – 420mm by 297mm

a6paper Sets \RM@papertype to 16 – 148mm by 105mm

a7paper Sets \RM@papertype to 17 – 105mm by 74mm

a8paper Sets \RM@papertype to 18 – 74mm by 52mm

a9paper Sets \RM@papertype to 19 – 52mm by 37mm

a10paper Sets \RM@papertype to 20 – 37mm by 26mm

b0paper Sets \RM@papertype to 30 – 1414mm by 1000mm

b1paper Sets \RM@papertype to 31 – 1000mm by 707mm

b2paper Sets \RM@papertype to 32 – 707mm by 500mm

b3paper Sets \RM@papertype to 33 – 500mm by 353mm

b4paper Sets \RM@papertype to 34 – 353mm by 250mm

b6paper Sets \RM@papertype to 36 – 176mm by 125mm

b7paper Sets \RM@papertype to 37 – 125mm by 88mm

b8paper Sets \RM@papertype to 38 – 88mm by 62mm

b9paper Sets \RM@papertype to 39 – 62mm by 44mm

b10paper Sets \RM@papertype to 40 – 44mm by 31mm

c0paper Sets \RM@papertype to 50 – 1297mm by 917mm

c1paper Sets \RM@papertype to 51 – 917mm by 648mm

c2paper Sets \RM@papertype to 52 – 648mm by 458mm

c3paper Sets \RM@papertype to 53 – 458mm by 324mm

c4paper Sets \RM@papertype to 54 – 324mm by 229mm

c5paper Sets \RM@papertype to 55 – 229mm by 162mm

c7paper Sets \RM@papertype to 57 – 114mm by 81mm

c7/6paper Sets \RM@papertype to 59 – 162mm by 81mm

bspopseedenvelopepaper Sets \RM@papertype to 60 – 152mm by 102mm. BS4264 UK post office preferred envelope: seed packets, wage slips, general packaging. The name is one I invented.

bspopnonisoenvelopepaper Sets \RM@papertype to 61 – 229mm by 102mm. BS4264 UK post office preferred envelope: gen commercial, non iso sizes. The name is one I invented.

bsbrochureenvelopepaper Sets \RM@papertype to 62 – 254mm by 178mm. BS4264 envelope; bulky A5, catalogues, brochures. The name is one I invented.

bslegalenvelopepaper Sets \RM@papertype to 63 – 270mm by 216mm. BS4264 envelope; legal docs, catalogues, photos. The name is one I invented.

bslargelegalenvelopepaper Sets \RM@papertype to 64 – 305mm by 127mm. BS4264 envelope; insurance policies, legal docs. The name is one I invented.

bscalendarenvelopepaper Sets \RM@papertype to 65 – 381mm by 254mm. BS4264 envelope; bulky docs, calendars. The name is one I invented.

foolscapfoliopaper Sets \RM@papertype to 70 – 13.5in by 8.5in

foolscappaper Sets \RM@papertype to 70 – 13.5in by 8.5in
foolscapquartopaper Sets \RM@papertype to 71 – 8.5in by 6.75in
foolscapoctavopaper Sets \RM@papertype to 72 – 6.75in by 4.25in
crownfoliopaper Sets \RM@papertype to 73 – 15in by 10in
crownquartopaper Sets \RM@papertype to 74 – 10in by 7.5in
crownoctavopaper Sets \RM@papertype to 75 – 7.5in by 5in
royalfoliopaper Sets \RM@papertype to 76 – 20in by 12.5in
royalquartopaper Sets \RM@papertype to 77 – 12.5in by 10in
royaloctavopaper Sets \RM@papertype to 78 – 10in by 6.25in
imperialfoliopaper Sets \RM@papertype to 79 – 22in by 15.5in
imperialquartopaper Sets \RM@papertype to 80 – 15in by 11in
imperialoctavopaper Sets \RM@papertype to 81 – 11in by 7.5in
largecrownoctavopaper Sets \RM@papertype to 82 – 8in by 5.25in
demyoquartopaper Sets \RM@papertype to 83 – 11.25in by 8.75in
demyoctavopaper Sets \RM@papertype to 84 – 8.75in by 5.625in
mediumquartopaper Sets \RM@papertype to 85 – 12in by 9.5in
mediumoctavopaper Sets \RM@papertype to 86 – 9.5in by 6in
ra0paper Sets \RM@papertype to 90 – 1270mm by 960mm
ra1paper Sets \RM@papertype to 91 – 1270mm by 960mm
ra2paper Sets \RM@papertype to 92 – 1270mm by 960mm
sra0paper Sets \RM@papertype to 93 – 1280mm by 900mm
sra1paper Sets \RM@papertype to 94 – 900mm by 840mm
sra2paper Sets \RM@papertype to 95 – 640mm by 450mm
metricdoublecrownpaper Sets \RM@papertype to 96 – 770mm by 505mm
metricquadcrownpaper Sets \RM@papertype to 97 – 1010mm by 770mm
metriclargequadcrownpaper Sets \RM@papertype to 98 – 1060mm by 820mm
metricquaddemypaper Sets \RM@papertype to 99 – 1030mm by 890mm
metricsmallquadroyalpaper Sets \RM@papertype to 100 – 1270mm by 960mm

The long paper sizes are described in detail in section 5.1.2 on page 46.

notlongpaper Sets \RM@longpapertypelong to 0; not long—the default.
7/8longpaper Sets \RM@longpapertypelong to 1; 7/8 long. The selected paper size has its longest dimension multiplied by 7/8.
3/4longpaper Sets \RM@longpapertypelong to 2; 3/4 long. The selected paper size has its longest dimension multiplied by 3/4.
2/3longpaper Sets \RM@longpapertypelong to 3; 2/3 long. The selected paper size has its longest dimension multiplied by 2/3.
5/8longpaper Sets \RM@longpapertypelong to 4; 5/8 long. The selected paper size has its longest dimension multiplied by 2/3.
1/2longpaper Sets \RM@longpapertypelong to 5; 1/2 long. The selected paper size has its longest dimension multiplied by 1/2. This is slightly different for asking for the next size down in an ISO series; these long sizes are not rounded to the nearest millimetre, as are standard ISO paper sizes, and code which sets things up for particular printer/paper combinations does not recognize 1/2 long A3 as A4 (for example).
3/8longpaper Sets \RM@longpapertypelong to 6; 3/8 long. The selected paper size has its longest dimension multiplied by 3/8.

`1/3longpaper` Sets `\RM@longpapertypelong` to 7; 1/3 long. The selected paper size has its longest dimension multiplied by 1/3.

`1/4longpaper` Sets `\RM@longpapertypelong` to 8; 1/4 long. The selected paper size has its longest dimension multiplied by 1/4.

`1/8longpaper` Sets `\RM@longpapertypelong` to 9; 1/8 long. The selected paper size has its longest dimension multiplied by 1/8.

5.2.2 Marginal paragraph options

The width of a marginal paragraph is set to the space left in the appropriate margin, taking into account all the limits. `rmpage` thinks the appropriate margin is this: in the case of multi-column printing, the smallest margin; in the case of one sided printing, normal marginal paragraph placement, in the outside margin; in the case of one sided printing, reverse marginal paragraph placement, in the inside margin; in the case of two sided printing, normal marginal paragraph placement, in the outside margin; and in the case of two sided printing, reverse marginal pragraph placement, in the inside margin.

The size is calculated on this basis: the standard LaTeX length `\marginparsep` gives the space between the text body and the marginal paragraph. The new length `\RM@mparclearance` gives the minimum space between the outside edge of the marginal paragraph and the edge of the paper (subject to the additional restrictions of `\RM@minrightclearance` and `\RM@minleftclearance` (but not `\RM@mininsidemargin` or `\RM@minoutsidemargin`; these apply to the text body only). Within these limits, `\marginparwidth` cannot be set to greater than the length `\RM@maxmparwidth`.

This way of setting marginal paragraphs is derived from the standard LaTeX 2_ε method, which uses 2 in as the largest allowed size, and 0.4 in as the minimum gap to the edge of the paper. `rmpage`'s equivalent parameters

You can scale the size of `\marginparsep`, `\RM@mparclearance`, and `\RM@maxmparwidth` using the `mparsep`, `mparclearance`, and `maxmparwidth` option sets. Look at section 3.7 on page 33 for more on marginal paragraphs.

If you think that the base value of any of these lengths is too small, you can do something about it. With `\marginparsep`, you could use the `\setlength` command to set it to a different value before loading `rmpage`. For example,

```
\setlength{\marginparsep}{2\marginparsep}
\usepackage{rmpage}
```

Because the other two parameters are given their initial values in `rmpage`, this technique won't work. The initial values of `\RM@mparclearance` and `\RM@maxmparwidth` are calculated as a certain fraction of `\paperwidth`; the initial value of the appropriate parameter is doubled if you specify the `largebasemparclear` or `largebasemaxmparwidth` options. You can set either of these parameters in the configuration file—the values -666 pt and -667 pt are reserved by `rmpage` as flag values; any positive length that's not too long is okay. Read the source and consider setting these parameters on a class-by-class basis if you do need to change them.

The `\mparsep` options scale the `\marginparsep` length using `\RM@scalebyoption`. See section 6.7 on page 77 for the details.

`leastmparsep` Sets `\RM@mparsepoption` to 3
`lessmparsep` Sets `\RM@mparsepoption` to 6
`lessishmparsep` Sets `\RM@mparsepoption` to 9
`normalmparsep` Default. Sets `\RM@mparsepoption` to 12
`moreishmparsep` Sets `\RM@mparsepoption` to 15
`moremparsep` Sets `\RM@mparsepoption` to 18
`mostmparsep` Sets `\RM@mparsepoption` to 21

The `...basemparclear` options need to be executed after `\paperwidth` has been set. Easily done with `\ProcessOptions` rather than `\ProcessOptions*`, and the papersize setting options declared above rather than below. The `\smallbasemparclear` value is set after option processing if no other value has been set. If `\RM@mparclearance` is -666 pt, the normalbasemparclear value is set; if it's -667 pt, the largebaselinemparclear value is set. The larger value is double the smaller value. This setting is done just after the `\RM@PrinterPaperSettings` hook is executed, which is well after `\paperwidth` is set. This value is scaled by option (using the `\RM@mparclearoption` passed to the `\RM@scalebyoption` command) just before it's used, so one can use the `\RM@BeforeWidthSetting` hook to change things.

`normalbasemparclear` Default. Sets `\RM@mparclearance` to -666pt.
`largebasemparclear` Sets `\RM@mparclearance` to -667pt
`normalbasemaxmparwidth` Default. Sets `\RM@maxmparwidth` to -666pt
`largebasemaxmparwidth` Sets `\RM@maxmparwidth` to -667pt

The gap between the edge of the paper and the edge of a marginal paragraph is 0.4 in (10.16 mm) with LaTeX's standard classes. `rmpage` changes this by introducing a new parameter, `\RM@mparclearance`, which is calculated as a fraction of `\paperwidth`. Normal `\RM@mparclearance` with A4 portrait paper is 9.88 mm; 0.4 in with US letter paper).

Touch options for `\RM@mparclearance` and `\RM@maxmparwidth` have been added now.

`leastmparclearance` Sets `\RM@mparclearoption` to 3
`lessmparclearance` Sets `\RM@mparclearoption` to 6
`lessishmparclearance` Sets `\RM@mparclearoption` to 9
`normalmparclearance` Default. Sets `\RM@mparclearoption` to 12
`moreishmparclearance` Sets `\RM@mparclearoption` to 15
`moremparclearance` Sets `\RM@mparclearoption` to 18
`mostmparclearance` Sets `\RM@mparclearoption` to 21

`\RM@maxmparwidth` is set as a fraction of `\paperwidth`, such that with portrait US letter paper, you get 2 in as standard, just like the standard classes. It's scaled by option (see section 6.7 on page 77 for the details). If you want to change the default base value of `\RM@maxmparwidth`, the `\RM@BeforeWidthSetting` hook is an ideal place to do it.

`leastmaxmparwidth` Sets `\RM@maxmparwidthoption` to 3
`lessmaxmparwidth` Sets `\RM@maxmparwidthoption` to 6
`lessishmaxmparwidth` Sets `\RM@maxmparwidthoption` to 9

`normalmaxmparwidth` Default. Sets `\RM@maxmparwidthoption` to 12
`moreishmaxmparwidth` Sets `\RM@maxmparwidthoption` to 15
`moremaxmparwidth` Sets `\RM@maxmparwidthoption` to 18
`mostmaxmparwidth` Sets `\RM@maxmparwidthoption` to 21

5.2.3 Touch options

All the `touch` options add or subtract one from a counter that is used to control the size of a page layout parameter. The effect of this is to give the layout parameter a size in between the 'main' sizes. That is, if you ask for `wide` and `touchwider`, `\textwidth` is set to a value 1/3 of the way (in a geometrical sequence) from `wide` to `wider`. This results in even step sizes: `wide`, `touchwider`, and `t@uchwider` give a width the same as `wider` and `touchnarrower`.

The `touch` options are intended to be used in documents; the `t@uch` options are intended to be used in class files. The reason is this: a class designer can develop a suitable layout by passing options to `rmpage`. When this is done, the options passed to `rmpage` can be passed using the `\PassOptionsToPackage` command in a class file. Any `touch` options should be turned into `t@uch` options, so that the controlled parameter can be incremented or decremented by a `touch` option in a document. Whether this is a good thing is unclear, but it's certainly deliberate.

Note that all the `touch` options need to be executed after their corresponding 'straight' options. To ensure this, none of the `touch` can be allowed in an `\ExecuteOptions` statement. The `\RM@notinexecuteoptions` is used in each of these option declarations: it produces an error message if used before the `\RM@donewithoptions` flag is set true, which is immediately before the `\ProcessOptions` statement.

You can find out more about the affected layout parameters by looking at the documentation for the main options corresponding to the `touch` options listed here.

`t@uchlonger` Adds 1 to `\RM@lengthoption`
`t@uchshorter` Adds -1 to `\RM@lengthoption`
`touchlonger` Adds 1 to `\RM@lengthoption`
`touchshorter` Adds -1 to `\RM@lengthoption`. See section 5.1.4.
`touchmorecolsep` Adds 1 to `\RM@columnsepoption`
`touchlesscolsep` Adds -1 to `\RM@columnsepoption`
`t@uchmorecolsep` Adds 1 to `\RM@columnsepoption`
`t@uchlesscolsep` Adds -1 to `\RM@columnsepoption`. See section 5.1.6.
`touchmoremparsep` Adds 1 to `\RM@mparsepoption`
`touchlessmparsep` Adds -1 to `\RM@mparsepoption`
`t@uchmoremparsep` Adds 1 to `\RM@mparsepoption`
`t@uchlessmparsep` Adds -1 to `\RM@mparsepoption`. See section 5.2.2.
`touchmorefootskip` Adds 1 to `\RM@footskipoption`
`touchlessfootskip` Adds -1 to `\RM@footskipoption`
`t@uchmorefootskip` Adds 1 to `\RM@footskipoption`
`t@uchlessfootskip` Adds -1 to `\RM@footskipoption`. See section 5.1.12.
`touchmoreheadsep` Adds 1 to `\RM@headsepoption`

`touchlessheadsep` Adds -1 to `\RM@headsepoption`
`t@uchmoreheadsep` Adds 1 to `\RM@headsepoption`
`t@uchlessheadsep` Adds -1 to `\RM@headsepoption`. See section 5.1.12.
`t@uchwider` Adds 1 to `\RM@widthoption`
`t@uchnarrower` Adds -1 to `\RM@widthoption`
`touchwider` Adds 1 to `\RM@widthoption`
`touchnarrower` Adds -1 to `\RM@widthoption`. See section 5.1.7.
`t@uchmoreoffset` Adds 1 to `\RM@offsetoption`
`t@uchlessoffset` Adds -1 to `\RM@offsetoption`
`touchmoreoffset` Adds 1 to `\RM@offsetoption`
`touchlessoffset` Adds -1 to `\RM@offsetoption`. See section 5.1.9.
`t@uchhigher` Adds -1 to `\RM@headfootbalance`
`t@uchlower` Adds 1 to `\RM@headfootbalance`
`touchhigher` Adds -1 to `\RM@headfootbalance`
`touchlower` Adds 1 to `\RM@headfootbalance`. See section 5.1.13.
`t@uchlessmparclearance` Adds -1 to `\RM@mparclearoption`
`t@uchmoremparclearance` Adds 1 to `\RM@mparclearoption`
`touchlessmparclearance` Adds -1 to `\RM@mparclearoption`
`touchmoremparclearance` Adds 1 to `\RM@mparclearoption`.
See section 5.2.2.
`t@uchlessmaxmparwidth` Adds -1 to `\RM@maxmparwidthoption`
`t@uchmoremaxmparwidth` Adds 1 to `\RM@maxmparwidthoption`
`touchlessmaxmparwidth` Adds -1 to `\RM@maxmparwidthoption`
`touchmoremaxmparwidth` Adds 1 to `\RM@maxmparwidthoption`.
See section 5.2.2.

5.2.4 More length options

These options need to be executed after the `touchlength` options; otherwise, they'd be in `rmpage.sty`.

Most of the `\textheight` setting options are in `rmpage` proper; see 5.1.4 on page 49.

`fulllength` Sets `\RM@lengthoption=30`. Makes `\textheight` as long as possible, taking into account the various restrictions and the need to keep `\textheight` to an integer number times `\baselineskip` plus `\topskip`.

`stdlength` Sets `\RM@lengthoption=0`. Makes `\textheight` the size it would be if you were using the standard LaTeX classes. I'm not sure I've checked everything that needs to be checked to ensure that this option always does what you'd expect, but I think I have. Note that `rmpage` still takes notice of footers and headers when you use this option, so you can get the standard `\textheight` with different vertical positioning of the text body.

5.2.5 Number of columns

If you are typesetting text in more than one column, `rmpage` needs to know so that it can set `\textwidth` appropriately.

The standard LaTeX options `onecolumn` and `twocolumn` are recognized by `rmpage`.

The `...columnwidth` options tell `rmpage` that you will be typesetting your text in that number of columns, but the only effect of the options is to change the `\textwidth` calculation. If you want to change the number of columns that your text is set in, you can use a package like multicol.

> These options affect only the character-based `\textwidth` calculation—for any given width option (`normalwidth`, `narrower`, or whatever) one text column is allowed to be a certain number of characters wide. Use the `...columnwidth` options to tell `rmpage` how many columns wide your text is, so it can calculate `\textwidth` appropriately.
>
> `rmpage` makes the character-based `\textwidth` guess equal to a certain number of characters times the number of columns, plus `\columnsep` times one less than the number of columns.

If your text is something like a single large table (for example, a timetable), it might be more appropriate to use the `paperwidthset` option to set `\textwidth` using the paper-based `\textwidth` only.

`onecolumnwidth` Default. Defines `\RM@textcols` to 1
`twocolumnwidth` Defines `\RM@textcols` to 2
`threecolumnwidth` Defines `\RM@textcols` to 3
`fourcolumnwidth` Defines `\RM@textcols` to 4
`fivecolumnwidth` Defines `\RM@textcols` to 5
`sixcolumnwidth` Defines `\RM@textcols` to 6
`sevencolumnwidth` Defines `\RM@textcols` to 7
`eightcolumnwidth` Defines `\RM@textcols` to 8
`ninecolumnwidth` Defines `\RM@textcols` to 9
`tencolumnwidth` multicol's limit. Defines `\RM@textcols` to 10

5.2.6 Loading founts

All the options below change how `\textwidth` is set, and the options with `load` in their name also call one of the standard PSNFSS packages to load the fount (aside from Lucida Casual and Concrete, which aren't PSFNSS founts, and Courier, which is handled anomalously).

> The only Lucida typeface I have is Lucida Casual, so that's the only Lucida typeface that `rmpage` deals with explicitly. If you do use others, you can have an appropriate `\textwidth` set using the `thisfountwidth` option.
>
> Given that the PSFNSS distribution has support for all the Lucida founts, I could be persuaded to include explicit support for them in `rmpage` if anyone's interested.

Remember that `\textwidth` is usually set to be a certain number of character wide. Well, not all founts have the same number of characters per inch. If you say `timeswidth` to `rmpage`, it will calculate a `\textwidth` based on the measured average width of one character in Times of the specified size.

This means that while the standard LaTeX classes would give you a `\textwidth` that is far too wide for the Times fount (which is generally narrower

than Computer Modern Roman), rmpage will give you a \textwidth that is pretty much the same number of characters across, which means you retain good legibility (as well as similar line and page breaks).

If you say loadtimes, rmpage changes its \textwidth calculation, loads the PSNFSS package that loads the Times fount family, and (very important, this is) changes the typesetting parameters to similar values to the ones suggested by Karl Berry, the chap who wrote the fontinst program that generated the virtual founts used to typeset the PSNFSS founts.

If you specify a looseness option yourself—see section 5.1.3 on page 48—it will over-ride the standard looseness set by a loadfount option. The loadconcrete option requests standard tight typesetting, and is anyway not recommended: if you want to use the concrete founts, try the beton package, which does a very good job of setting up LaTeX to use these founts. rmpage can work happily with beton: see section 5.1.15 on page 55.

You can tell rmpage to set the \textwidth based on the width of a fount it doesn't know about with the thisfountwidth option. If you use this option, rmpage will calculate a \textwidth based on the size of the fount that is current when rmpage is loaded. So if you want a \textwidth based on, say, Grunge Update (family name fgr on my computer), you could say:

```
\documentclass[thisfountwidth,12pt]{article}
\renewcommand{\rmdefault}{fgr}
\rmfamily
\usepackage{rmpage}

\begin{document}

...
```

rmpage will tell you which fount it is working with, and the results of its calculations. If you get confused by LaTeX's fount selection scheme, read the manual; it confuses me too.

The fount options work like this: each option sets the \RM@fountfamily command to a particular value. Any option which sets the \RM@loadfount flag true forces code later on in rmpage to load the appropriate fount, most of them using one of the standard PSNFSS packages. The fount loading code is written specially for each fount; there's no easy way to add more founts to the list that's already dealt with. But you could add code to the \RM@AfterProcessOptions hook if you want to do this; I suggest that this code loads the fount, selects it, and defines \RM@fountfamily to be 12, to make the width setting code measure its width. There are no hooks in rmpwnorm.pko to add this sort of thing.

> The thing about using the loadfount options is that the standard PSNFSS packages don't always set the default main document typeface (the one you get when you ask for \rmfamily) to be the fount you've asked for. So if you say loadhelvet, you'll get a \textwidth based on the width of the Helvetica typeface, but Helvetica is the fount you get when you ask for \sffamily, which might not be what you want.

> The thing to do is be sure which typeface will be the main document face, and ask `rmpage` to set the `\textwidth` accordingly. You might use the appropriate `loadfount` option for this, or load the founts you want with separate calls to the appropriate packages in your document's preamble.

If you don't have the file needed to load the requested fount family, `rmpage` complains.

Fount families are set like this:

```
0=cmr
1=avant garde 2=bookman 3=zapf chancery 4=helvetica
5=new century schoolbook 6=palatino 7=times 8=utopia
9=lucida casual 10=courier 11=concrete 12=this fount width
13=lucida casual dirty trick
```

The dirty trick works like this: if you ask for `loadlucidacasual`, rather than `loadluccasua` (`\RM@fountfamily` 13), the fount loading code later on does a `\RequirePackage{lucida-casual}`, and then sets `\RM@fountfamily` to 9. This lets Rowland use his own `.fd` files for Lucida Casual, and allows access to the standard `.fd` files. The option to do this is in Rowland's curious option section of the configuration file.

`cmrwidth` Computer modern roman; redundant options.
> Defines `\RM@fountfamily` to 0

`loadcmr` Loads nothing; does whinge a little.
`avantwidth` Avant Garde. Defines `\RM@fountfamily` to 1
`loadavant` Requires the avant package.
`bookmanwidth` Bookman. Defines `\RM@fountfamily` to 2
`loadbookman` Requires the bookman package.
`chancerywidth` Zapf Chancery. Defines `\RM@fountfamily` to 3
`loadchancery` Requires the chancery package.
`helvetwidth` Helvetica. Defines `\RM@fountfamily` to 4
`loadhelvet` Requires the helvet package.
`newcentwidth` New Century Schoolbook. Defines `\RM@fountfamily` to 5
`loadnewcent` Requires the newcent package.
`palatinowidth` Palatino. Defines `\RM@fountfamily` to 6
`loadpalatino` Requires the palatino package.
`timeswidth` Times. Defines `\RM@fountfamily` to 7
`loadtimes` Requires the times package.
`utopiawidth` Utopia. Defines `\RM@fountfamily` to 8
`loadutopia` Requires the utopia package.
`lucasualwidth` Lucida casual. Defines `\RM@fountfamily` to 9
`loadlucasual` Requires the lucasual package.
`courierwidth` Courier. Defines `\RM@fountfamily` to 10
`loadcourier` This option makes the default roman fount Courier. I think this is ugly and crude: you might be better off using the times package and `\ttfamily`
`concretewidth` Concrete. Defines `\RM@fountfamily` to 11
`loadconcrete` This option loads the beton package and sets `\textwidth` for the Concrete Roman founts.

`thisfountwidth` Bases `\textwidth` on the current fount.
Defines `\RM@fountfamily` to 12

5.2.7 Stuff for beton support

`standard-baselineskips` Passes this option to beton so `rmpage` can detect whether this option's been passed to beton.
`oldstyle-equation-numbers` Passes this option to beton.
`concrete-math` Passes this option to beton.

5.2.8 Other synonyms for some options

Why oh why oh why do I have to make my package speak US English? (mutter mumble grumble). Yes, all right, it's how the convention's worked out. And I've put the original options to change the `\today` command here too. (It's like this: I once had a LaTeX 2.09 style file called nicedate that changed the `\today` command into something I liked. Then I wrote `rmpage`, and included the nicedate code, activated by the `nicedate` option. Obviously, the `nicedate` option needed a complementary option, and the obvious name for this option was `nastydate`, which eventually got turned into `othernicedate`. After I started turning `rmpage` into something for the rest of the world to look at (which it wasn't originally), I added the `ukdate` and `usdate` options. But I still like `nicedate`, so I've kept it. So there.)

`othernicedate` The same as `usdate`; does nothing
`nicedate` The same as `ukdate`; changes date format.
`verbose` Synonym for chatty
`silent` Synonym for yorkshire
`errorshow` Synonym for yorkshire
`warningshow` Synonym for taciturn
`infoshow` Synonym for chatty
`debugshow` Synonym for garrulous
`center` The same as `centre`
`notcenter` The same as `notcentre`

5.2.9 Margin options

`ringbinding` This option sets the minimum allowed inside margin to be at least 15 mm if you are printing in portrait orientation. It's in the config file because it must be executed after the `landscape` and `portrait` options. It does nothing but warn you if you use it in landscape orientation.

Beware that this option takes no notice of long paper sizes at all, and is likely to give iffy results if you combine it with them. If you have any thoughts about this option, please email me—I'm not terribly happy with it.

5.2.10 Printer options

Each printer option must set these ten parameters:

`\RM@printertype` A code number, defined below. This number is used by `rmpage` to keep track of the printer used; you can tell `rmpage` to do things for certain printers and not for others.

`\RM@ptrrportclear` The non-printing margin on the right-hand side in portrait orientation

`\RM@ptrlportclear` The non-printing margin on the left-hand side in portrait orientation

`\RM@ptrtportclear` The non-printing margin at the top in portrait orientation

`\RM@ptrbportclear` The non-printing margin at the bottom in portrait orientation

`\RM@ptrrlandclear` The non-printing margin on the right-hand side in landscape orientation

`\RM@ptrllandclear` The non-printing margin on the left-hand side in landscape orientation

`\RM@ptrtlandclear` The non-printing margin at the top in landscape orientation

`\RM@ptrblandclear` The non-printing margin at the bottom in landscape orientation

`\RM@ptrpostol` Nominally, the amount you expect the position of the paper to vary. The value of this command is added to each of the `\ptr...clear` parameters before they are used.

Printer types are:

0=fullbleed 1=general 2=pessimistic 3=optimistic
10=dw300 11=dw500 12=dw600 (HP deskwriter inkjet series)
20=lj2 21=lj3 22=lj4 (HP laserjet laser printer series)
30=canonbjx (Canon bubblejet flurble)
40+ others (whatever comes up)
1000+ local printers to avoid clashes

The figures for all these printers are guesses, except for the DW500 and DW600: any data on printing margins for the printers above or other commonly-used printers would be gratefully received. I need to know about printing limits at the top, bottom, left, and right for portrait and landscape modes, and whether the data is what the book says or what you measured (preferrably both, but anything'll help). If anyone really uses LaTeX with an A3 printer, do tell: it's something I've been wondering about.

When I've got a better idea of what's going on, I'll define more printer options.

`\RM@ptrpostol` generally set to 1 mm (paper sizes are to ±2 mm), except for our DW520 which I keep a careful eye on.

`fullbleedprinter` Lets you print all the way to the edge of the paper.

```
\def\RM@printertype{0}
\def\RM@ptrrportclear{0mm}  \def\RM@ptrrlandclear{0mm}
\def\RM@ptrlportclear{0mm}  \def\RM@ptrllandclear{0mm}
```

```
\def\RM@ptrtportclear{0mm}  \def\RM@ptrtlandclear{0mm}
\def\RM@ptrbportclear{0mm}  \def\RM@ptrblandclear{0mm}
\def\RM@ptrpostol{0mm}
```

`generalprinter` Arbitrary settings that probably ensure a layout inside the printing area on most A4 printers.

```
\def\RM@printertype{1}
\def\RM@ptrrportclear{8mm}    \def\RM@ptrrlandclear{8mm}
\def\RM@ptrlportclear{8mm}    \def\RM@ptrllandclear{8mm}
\def\RM@ptrtportclear{8mm}    \def\RM@ptrtlandclear{8mm}
\def\RM@ptrbportclear{15mm}   \def\RM@ptrblandclear{15mm}
\def\RM@ptrpostol{1mm}
```

`pessimisticprinter` This uses the worst limits I can remember meeting, so it force documents inside the printing area on any printer.

```
\def\RM@printertype{2}
\def\RM@ptrrportclear{10mm}  \def\RM@ptrrlandclear{19mm}
\def\RM@ptrlportclear{10mm}  \def\RM@ptrllandclear{10mm}
\def\RM@ptrtportclear{10mm}  \def\RM@ptrtlandclear{10mm}
\def\RM@ptrbportclear{19mm}  \def\RM@ptrblandclear{10mm}
\def\RM@ptrpostol{1mm}
```

`optimisticprinter` This uses the best limits I'd expect from a real printer.

```
\def\RM@printertype{3}
\def\RM@ptrrportclear{3mm}  \def\RM@ptrrlandclear{3mm}
\def\RM@ptrlportclear{3mm}  \def\RM@ptrllandclear{3mm}
\def\RM@ptrtportclear{3mm}  \def\RM@ptrtlandclear{3mm}
\def\RM@ptrbportclear{3mm}  \def\RM@ptrblandclear{3mm}
\def\RM@ptrpostol{0.5mm}
```

`dw300printer` A guess.

```
\def\RM@printertype{11}
\def\RM@ptrrportclear{6mm}  \def\RM@ptrrlandclear{15mm}
\def\RM@ptrlportclear{6mm}  \def\RM@ptrllandclear{7mm}
\def\RM@ptrtportclear{7mm}  \def\RM@ptrtlandclear{6mm}
\def\RM@ptrbportclear{15mm} \def\RM@ptrblandclear{6mm}
\def\RM@ptrpostol{1mm}
```

`dw500printer` Hewlett-Packard's specification for its DeskWriter and DeskJet 500/510/520/540 printers.

```
\def\RM@printertype{11}
\def\RM@ptrrportclear{6mm}  \def\RM@ptrrlandclear{15mm}
\def\RM@ptrlportclear{6mm}  \def\RM@ptrllandclear{7mm}
\def\RM@ptrtportclear{7mm}  \def\RM@ptrtlandclear{6mm}
\def\RM@ptrbportclear{15mm} \def\RM@ptrblandclear{6mm}
\def\RM@ptrpostol{1mm}
```

`dw600printer` Measured from a particular HP 600 series inkjet printer, with a bit added.

```
\def\RM@printertype{12}
\def\RM@ptrrportclear{5mm}  \def\RM@ptrrlandclear{15mm}
\def\RM@ptrlportclear{5mm}  \def\RM@ptrllandclear{2mm}
\def\RM@ptrtportclear{2mm}  \def\RM@ptrtlandclear{5mm}
\def\RM@ptrbportclear{15mm} \def\RM@ptrblandclear{5mm}
\def\RM@ptrpostol{1mm}
```

lj2printer an arbitrary guess

```
\def\RM@printertype{20}
\def\RM@ptrrportclear{7mm}  \def\RM@ptrrlandclear{7mm}
\def\RM@ptrlportclear{7mm}  \def\RM@ptrllandclear{7mm}
\def\RM@ptrtportclear{7mm}  \def\RM@ptrtlandclear{7mm}
\def\RM@ptrbportclear{7mm}  \def\RM@ptrblandclear{7mm}
\def\RM@ptrpostol{1mm}
```

lj3printer an arbitrary guess

```
\def\RM@printertype{21}
\def\RM@ptrrportclear{6mm}  \def\RM@ptrrlandclear{6mm}
\def\RM@ptrlportclear{6mm}  \def\RM@ptrllandclear{6mm}
\def\RM@ptrtportclear{6mm}  \def\RM@ptrtlandclear{6mm}
\def\RM@ptrbportclear{6mm}  \def\RM@ptrblandclear{6mm}
\def\RM@ptrpostol{1mm}
```

lj4printer an arbitrary guess

```
\def\RM@printertype{22}
\def\RM@ptrrportclear{5mm}  \def\RM@ptrrlandclear{5mm}
\def\RM@ptrlportclear{5mm}  \def\RM@ptrllandclear{5mm}
\def\RM@ptrtportclear{5mm}  \def\RM@ptrtlandclear{5mm}
\def\RM@ptrbportclear{5mm}  \def\RM@ptrblandclear{5mm}
\def\RM@ptrpostol{1mm}
```

canonbjxprinter an arbitrary guess

```
\def\RM@printertype{22}
\def\RM@ptrrportclear{7mm}  \def\RM@ptrrlandclear{12mm}
\def\RM@ptrlportclear{7mm}  \def\RM@ptrllandclear{7mm}
\def\RM@ptrtportclear{7mm}  \def\RM@ptrtlandclear{7mm}
\def\RM@ptrbportclear{12mm} \def\RM@ptrblandclear{7mm}
\def\RM@ptrpostol{1mm}
```

5.2.11 Rowland's curious options

These are curious options, defined by me (RJMM) to perform dark and eldritch deeds. These aren't intended for hoi polloi, mainly 'cos they're a bit iffy in places, but I like them and they might give you some ideas.

Our DW520 isn't quite to spec.

```
R+R-dw520printer \def\RM@printertype{2}
     \def\RM@ptrrportclear{7mm}  \def\RM@ptrrlandclear{15mm}
```

```
\def\RM@ptrlportclear{6mm}  \def\RM@ptrllandclear{7mm}
\def\RM@ptrtportclear{7mm}  \def\RM@ptrtlandclear{7mm}
\def\RM@ptrbportclear{15mm} \def\RM@ptrblandclear{6mm}
\def\RM@ptrpostol{0.5mm}
```

`lucidacasualwidth` Lucida casual Defines `\RM@fountfamily` to 9

`loadlucidacasual` A dirty trick to load my .fd version of lucida casual rather than the PSNFSS version. If you `loadlucidacasual`, `\RM@fountfamily` is set to 9 after the lucida-casual package has been `\RequirePackage`d. That's done by code further on in rmpage, specially written for this dirty trick.

I have written packages that do the same job as the standard `size10.clo` etc., files, but for larger sizes. Because the standard LaTeX `\@ptsize` parameter is intended to be a single digit, and I want to use several different sizes, I have defined a new parameter that holds the point size of the main body type, the command `\RM@ptsize`. This parameter is only defined for my larger sizes.

Because my size packages can be loaded before or after `rmpage`, and because both need to know about the point size, `rmpage` says

```
\providecommand{\RM@ptsize}{666}
```

before the extra size options are executed. The options set `\RM@ptsize` to the appropriate value if this hasn't already been done. All my other packages that recognize the larger point size options do something similar.

The other thing that `rmpage` does with these larger point size options is set `\@ptsize` to the 12 pt value; this is to fool sections of `rmpage` into thinking that it's dealing with a 12 pt fount.

`14pt` Sets `\RM@ptsize` to `14` if needed and sets `\@ptsize` to 2 (meaning 12 12, to fool `rmpage`)

`24pt` Sets `\RM@ptsize` to `24` if needed and sets `\@ptsize` to 2 (meaning 12 12, to fool `rmpage`)

`36pt` Sets `\RM@ptsize` to `36` if needed and sets `\@ptsize` to 2 (meaning 12 12, to fool `rmpage`)

Chapter 6

How things work

6.1 \textheight calculation

The way rmpage decides on the value of \textheight is this: the length options set a length called \RM@totalheadfootclearance to be certain fraction of the \paperheight. The value of this command will be the sum of the blank space above the header and below footer, after it has been checked against several restrictions.

> If the noheaders option has used, \headheight and \headsep are both set to 0 pt; if the nofooters option has been used, \footskip is set to 0 pt. So a given length option will fill the page to the same extent whether or not headers or footers are used; turning headers and footers off will increase \textheight.

The first check that's made is that the \textheight produced by this value of \RM@totalheadfootclearance will not exceed the bounds set by \RM@mintextheight and \RM@maxtextheight. These two commands are intended to be set by local code on a class-by-class basis in the configuration file, using the \RM@OnClassType command in the \RM@AfterProcessOptions hook, and define the allowed range of \textheight. This check defines the commands:

\RM@maxpractextheight and \RM@mintotalheadfootclearance
\RM@minpractextheight and \RM@maxtotalheadfootclearance

They are calculated from the user supplied limits; they are based on the largest and smallest values that \textheight is allowed to have, given the limits of the discrete values it is allowed to take. The \practextheight parameters are unused at the moment; they might come in handy one day.

The total space above and below the footer is divided into two: a certain fraction of this space to the gap at the top, the rest to the gap at the bottom. These two lengths are saved in \RM@totalheadclearance and \RM@totalfootclearance. The altitude options control this division of space—see section 5.1.13 on page 55.

Checks are then made to ensure that nothing will be printed outside the allowed printing region along the vertical axis. This is defined by the lengths \RM@minheadclearance and \RM@minfootclearance, which are initially set

to the non-printing margin top and bottom. rmpage ensures that these two lengths are at least as large as either \RM@mintopmargin or \RM@minbottommargin, as appropriate—these two parameters are intended to be set class-by-class, so you can ensure that your layout meets regulations, for example.

rmpage issues warnings if it decides to make either the top or bottom gap larger to fit the text inside the printing region. The warnings are issued because changing the top or bottom gap changes the \textheight and vertical position of the text on the page; you might have been expecting a smooth increment from the last altitude or \length option, or a particular balance of space above and below the text given by a particular altitude option, and it might be useful to know that you've not got what you might expect.

Finally, \textheight is set to a value such that:

$$\texttt{\textheight} = \text{integer} \times \texttt{\baselineskip} + \texttt{\topskip}$$

The values of the three \RM@total...clearance parameters are increased to match the reduction in the size of \textheight, and \topmargin is set to whatever it needs to be.

The apparently special case of stdlength is handled by setting \RM@totalheadfootclearance to a value that will yield the same \textheight as the standard classes; the book, article, report, and letter classes use one value; the slides class another. Have a look at rmpage.dtx and the standard class files to see how this is done. Note that rmpage gives you the same \textheight as the standard classes whether or not you are using headers or footers—the number of text body lines on the page is always the same (at least, it always has been in testing). There are exceptions to this: it is possible to ask rmpage to position a stdlength page on the paper in such a fashion that the text would end up outside the printable region. In this case, rmpage will issue a warning and reduce \textheight to fit inside the allowed area.

> The parameters that define the available printing region along the vertical axis are \RM@minheadclearance and \RM@minfootclearance. The values of these \RM@...clearance parameters are set printer by printer, and possibly paper-size by paper-size. rmpage ensure that they are at least as large as either RM@mintopmargin or \RM@minbottommargin, as appropriate.
>
> If the particular combination of printer and paper has set the flag \RM@jackup to be true, rmpage will lift the printing region to clear an over-large non-printing margin at the bottom of the page. This is useful for people with Hewlett-Packard inkjet printers. There is no similar facility for automatically lowering the printing region or shifting it sideways. This is because a larger than expected space at the bottom of the page is rarely a problem, but lower than normal or shifted sidedways is usually a problem. These effects may be achieved, but you have to do it by using options to have the specific effect you want.
>
> Each printer option must define the commands:
>
> ```
> \RM@ptrrportclear   \RM@ptrlportclear   \RM@ptrtportclear
> \RM@ptrbportclear   \RM@ptrrlandclear   \RM@ptrllandclear
> \RM@ptrtlandclear   \RM@ptrblandclear   \RM@ptrpostol
> ```
>
> They define the non-printing clearances in landscape and portrait orientation, and the assumed maximum positional error. Code can be

added in the \RM@PrinterPaperSettings hook to set particular clearances for particular combinations of printer and paper (to cope with, for example, the 19 mm non-printing margin at the bottom of an envelope fed into an HP DeskWriter 520, which is much larger than the 15 mm non-printing margin at the bottom of a normal bit of paper).

The beton package changes \baselineskip, but does so at the start of the document, using the \AtBeginDocument hook provided by standard LaTeX. rmpage needs to know what the \baselineskip of the main document fount will be. To get round this problem, rmpage steals some code from beton so it can set \baselineskip to the value it will have after \AtBeginDocument. When the vertical page parameters have been calculated, rmpage puts \baselineskip back to its initial value.

None of this happens unless you load the beton package, and specify the beton option to rmpage. If you don't want rmpage to use beton's \baselineskip, specify the nobeton option to rmpage. Note that you must load the beton package before you load rmpage. Sorry.

6.2 \textwidth calculation

The way \textwidth is worked out is this: two different \textwidth initial guesses are calculated—one based on the width of a certain number of characters allowed in each column, the other based on a certain fraction of \paperwidth—and the smaller one is used as the basis for the final \textwidth. The particular number of characters and fraction of \paperwidth is set by the width option specified. The default normalwidth option gives a character-based width very close to standard, but the paper-based width is quite a bit larger than standard. This is only significant when rmpage uses the paper-based width, as it usually does when you are printing on A5 paper.

The technique of choosing the smaller of two \textwidths: one based on the number of characters, and the other based on the size of the paper, was derived from the standard classes' way of doing things—a fixed width (different for each size) is compared to \paperwidth − 2 in, and the smaller is used. The fixed width is different for each size; the conventional classes use hardwired sizes, and the slides class uses 65/2× width of (im).

rmpage asks for a normal text-based width based on the same number of characters as the standard widths, and compares this to a paper-based width that is calculated as 0.7138×\paperwidth. The fraction used gives a larger \textwidth than the standard classes when typesetting on A5 paper, but a smaller \textwidth when typesetting large founts on A4 paper.

The normal character-based \textwidth calculated by rmpage is different to that calculated by the standard classes because the standard classes round calculated dimensions down to the next lowest integer number of points. I think this is a mistake, because it introduces an unnecessary error in margin sizes, so rmpage doesn't do it (unless you're asking for stdwidth, which does truncates the calculated value of \textwidth only).

The special widths: oneinchmargins, halfinchmargins, and fullwidth, all set the paper-based textwidth to a fixed value to leave the specified amount of space either side, assuming that you have asked for centred printing (fullwidth leaves no space).If you haven't, the total space either

side will add up to what you'd expect, but you can get 1.5 in on one side, and 0.5 in on the other, for example, if you're using `oneinchmargins`.

`stdwidth` sets both the paper and character-based initial guesses to the same value as the standard classes. If the selected initial `\textwidth` value isn't reduced, the result will be the same as the standard classes' calculations (insert standard disclaimer here—not because I'm afraid of being sued, but because I think you should check that the value of `\textwidth` is what it should be if it's really important. This is because I think this is a complicated piece of software because TEX's piggin' awful for doing maths, and I've not verified the algorithm to my own satisfaction. I have tested it, and it appears to work the way I want, so that'll have to do for now.)

The first check ensures that this first guess is within the bounds of `\RM@mintextwidth` or `\RM@maxtextwidth`. If it's not, it's made big or small enough.

If you've asked for `characterwidthset`, then the paper-based `\textwidth` is set to a large value; similarly, if you've asked for `paperwidthset`, the character-based `\textwidth` is set to a large value. If you've asked for an inherently paper-based width like `oneinchmargins`, both the paper and character-based `\textwidth`s are set to the appropriate value. So if you also ask for `characterwidthset`, `rmpage` will give you `oneinchmargins` anyway, and a complaint.

Now the smallest of the two `\textwidth`s is selected as the one to use for real, and `rmpage` calculates `\evensidemargin` based on the requested offset—the proportions in which the available horizontal space is divided between the larger and smaller margins (see section 5.1.9 on page 52 for the details).

When `rmpage` checks `\textwidth`, it takes into account whether you're printing two sided or not. The checking code looks a bit complicated, because there are two sets of limits that apply to the horizontal extent of the text: `\RM@minrightclearance` and `\RM@minleftclearance`; and `\RM@minoutsidemargin` and `\RM@mininsidemargin`. `rmpage` looks at the appropriate limits.

The `\left` and `\right` `\minclearance` parameters define the possible printing region, as set for the requested printer and paper combination; all text must fit inside these limits. The `\inside` and `\outside` `\minmargin` parameters define the permitted extent of the main body text, excluding marginal paragraphs. The `\minmargin` parameters are defined to be `0pt` by default; they were introduced so that I could write a thesis class which had to ensure particular minimum margins to meet the regulations. You can see how I used them in the `\RM@AfterProcessOptions` hook definition in the configuration file.

If `\evensidemargin` is too small to allow the text to print on the page, it is increased, and `\textwidth` decreased to maintain the requested offset proportions.

Then the right-hand edge of `\textwidth` is checked to ensure that it is within the allowed printing region. If it's not, `\textwidth` is reduced. If so, `\evensidemargin` must be increased to maintain the requested offset proportions, and `\textwidth` reduced by the same amount to keep the right-hand margin the same size.

If you've asked for `\fullwidth`, `rmpage` won't attempt to retain the offset proportions, nor will it issue as many warnings about decreasing things to fit.

When that's done, `\oddsidemargin` is set to the appropriate value: equal to `\evensidemargin` if you've asked for one sided or centred printing; or equal to the right-hand margin on an even-numbered page if you're printing two sided and not centred.

And finally, `rmpage` checks that the final value of `\textwidth` is still larger than `\RM@mintextwidth`. If it's not, `rmpage` can't do anything about it, so just issues an error message.

6.3 Hooks

There's five hooks:

`\RM@BeforeProcessOptions`, `\RM@AfterProcessOptions`,
`\RM@PrinterPaperSettings`,
`\RM@BeforeWidthSetting`, `\RM@AfterWidthSetting`.

`\RM@BeforeProcessOptions` This hook is executed just before `\ProcessOptions`, and before the `\RM@donewithoptions` flag has been set to true, so options which can only be specified in an `\ExecuteOptions` statement can be requested.

`\RM@AfterProcessOptions` This hook is executed well after `\ProcessOptions`. It is executed after most of the fiddling about prior to working out page parameters has been done, just after the current class has been identified, but before class-specific code is executed. This is the hook to use if you want to add a new class: you should preferably set `\RM@classtype` with a new option, declared either directly in the config file, or by using the `\RM@BeforeProcessOptions` hook. Then put class-specific code in the `\RM@AfterProcessOptions` hook; use the `\RM@OnClassType` command.

`\RM@PrinterPaperSettings` This hook is executed after the standard printer/paper specific code has been executed. Use the `\RM@OnPrinterType`, `\RM@OnPortraitPaperType`, `\RM@OnLandscapePaperType`, and `\RM@OnPaperType` commands here.

`\RM@BeforeWidthSetting` If you want to use a different file for width setting, define `\RM@widthsetter` to be the name of the file in this hook. The `\RM@OnClassType` command can be used to select which class this file should be used for. This hook is executed after the `\RM@OnClassType` command has been set by the standard code, and just before the width setting file is loaded.

`\RM@AfterWidthSetting` This hook is executed on returning from the width setting file. It's here for æsthetic reasons.

6.4 Marginal paragraphs

There is more on setting the size of marginal paragraphs in section 6.7 on page 77.

If you are going to use marginal notes in your document, ensure that you specify `\normalmarginpar` or `\reversemarginpar` *before* loading `rmpage`. This is because `rmpage` calculates the size of marginal paragraphs based on the space available, and if `rmpage` thinks you're going to put marginal notes in the margin which is largest, and you really put them in the margin that is smallest, it'll get the calculation wrong and you'll have marginal notes that don't fit on the page.

If you're going to switch between `\normalmarginpar` and `\reversemarginpar` in your document, select whichever one puts the marginal notes in the smallest margin before you load `rmpage`. Messy, I know—sorry.

The standard LaTeX classes, report and article, create marginal paragraphs that are a fixed distance away from the text, with a minimum clearance from the edge of the paper of 0.8in (one sided printing), or 0.4in (two sided printing), and a maximum width of 2in.

`rmpage` says yah boo sucks to all this.

6.5 Dealing with different classes

`rmpage` gives each class a number; the number of the current class is stored in the command `\RM@classtype`. Classes are detected in `rmpage`, and in the config file. More than one class can have been loaded; the idea is that the first loaded class is defined as the current class.

6.6 Different paper types and printers

`rmpage` knows about non-printing margins, and about different paper sizes and orientations. Each printer has its own defined non-printing margins, which are used to calculate the non-printing margins for each paper size.

The non-printing margins for the selected paper size are calculated and stored in the commands:

`\RM@minrightclearance` `\RM@mintopclearance`
`\RM@minleftclearance` `\RM@minbottomclearance`

Each printer option must define nine commands so that `rmpage` can calculate the non-printing margins for each paper size. These commands to define the non-printing margins for each printer are:

Portrait orientation non-printing margins:
`\RM@ptrrportclear` right-hand side
`\RM@ptrlportclear` left-hand side
`\RM@ptrbportclear` bottom edge
`\RM@ptrtportclear` top edge
Landscape orientation non-printing margins: `\RM@ptrrlandclear` right-hand side

\RM@ptrllandclear left-hand side
\RM@ptrblandclear bottom edge
\RM@ptrtlandclear top edge
Assumed maximum positional error: \RM@ptrpostol

The \RM@ptrpostol command holds a length which is added to the \RM@minclearance values right at the end of the non-printing margin calculations, *after* the \RM@PrinterPaperSettings hook has been executed. Properly speaking, there should be four of these: one for left-right error and one for up-down error in both portrait and landscape orientation, but I think that one is probably adequate. The standard printer types set \RM@ptrpostol to 1 mm.

rmpage works out which printer clearance parameters to use as the non-printing margins like this;

1. The flag \RM@portrait is set true if you are printing in portrait orientation, false if you are printing landscape.
2. The flag \RM@portlandinvert is set true if you are using a long paper size which reduces the length of the parent paper size to less than the width of the parent paper size.
3. if (\RM@portrait and not \RM@portlandinvert)
   or (not \RM@portrait and \RM@portlandinvert)
   then use the portclear parameters
4. if (\RM@portrait and \RM@portlandinvert)
   or (not \RM@portrait and not \RM@portlandinvert)
   then use the landclear parameters
5. If you are using a long paper size that is greater than half the parent size, set to 0 pt the non-printing margin at the edge where you cut the parent size to create the long size (assumed to be either the right-hand or bottom edge).
6. Execute the \RM@PrinterPaperSettings hook
7. Add the value of the command \RM@ptrpostol to the non-printing margins

6.7 Headers, footers, and marginal paragraphs

This section looks at how to control the size of the gap between the main body text and: headers, footers, and marginal paragraphs, as well as the gap between marginal paragraphs and the edge of the paper, the maximum width of marginal paragraphs, and the gap between columns of text in a multiple-column layout.

The details of how the size of marginal paragraphs is calculated are in section 6.4 on page 76.

Some lengths used as page layout parameters are set to their final values by the command:

```
\RM@scalebyoption{<length to be scaled>}{<option number>}
```

The `<length to be scaled>` is multiplied by a factor controlled the the `<option number>`. The option number is set to 12 by default (which means multiply by one), and is allowed to range from 1 to 23. The multipliers are in a geometrical sequence from 0.3263 to 3.0646. The option `least...` sets the option number to 3, and gives a multiplier of 0.4. The option `most...` set the option number to 21, and gives a multiplier of 2.5. The `touchmore...` and `touchless...` options add or subtract one from the option number, as do the `t@uchmore...` and `t@uchless...` options (which are reserved for use by class files).

These following lengths are scaled by option; the name of the counter storing the controlling option number is given in each case.

1. `\headsep`—the gap between the top of the text and the box containing the head. Controlled by `\RM@headsepoption`.

2. `\footskip`—the gap between the bottom of the text and the baseline of the foot. `rmpage` scales the length (`\footskip` − `\baselineskip`), to scale the apparent gap between the (assumed one line) footer and the bottom of the text body. Controlled by `\RM@footskipoption`.

3. `\columnsep`—the gap between columns on a multi-column page (nothing to do with tables). Controlled by `\RM@columnsepoption`.

4. `\marginparsep`—the gap between the text body and marginal paragraphs. Controlled by `\RM@mparsepoption`.

5. `\RM@mparclearance`—the minimum gap between the edge of the paper and marginal paragraphs. Controlled by `\RM@mparclearoption`.

6. `\RM@maxmparwidth`—the maximum width of marginal paragraphs. Controlled by `\RM@maxmparwidthoption`.

The first four are standard LaTeX lengths, and the only change `rmpage` makes to them is with `\RM@scalebyoption`. The fifth and sixth lengths are new parameters set by `rmpage`.

You can play about with all of these parameters by passing options to `rmpage`.

If the spread of values given by the standard options isn't enough, you can say:

```
\setlength\columnsep{3\columnsep}
```

or some such in your preamble, **before** loading `rmpage`. This only works for the standard LaTeX parameters.

If you want to do something like this for `\RM@mparclearance` or `\RM@maxmparwidth`, you can use the `largebasemparclear` or `largebasemaxmparwidth` options. These multiply the corresponding parameter by 2 if it has not been set by a configuration file.

Chapter 7

Configuring rmpage

The configuration file exists so you can tailor your installation of `rmpage` to your preferences. Some obvious things to set are the default printer type, the default paper type, the default date style, whether you want the inside or the outside margin larger.

7.1 Setting up a new installation

`rmpage` will work entirely happily without local configuration, but you might want to customize its behaviour, for example to speed it up. This chapter explains how.

For the sake of compatibility with everyone else and future versions, please keep the file `rmpgen.cfg` unchanged in your TEX search path, and make changes only to a copy of one of the standard configuration files that came with `rmpage`. I suggest you make a copy called `rmplocal.cfg`, from either `rmpgen.cfg` (everything active) or `rmplocal.gfc` (fastest).

When you create your local configuration file, begin by doing two things: add a comment on the top line identifying this file as yours; and change the `\ProvidesFile` command to match the new name and identify the file as yours—don't forget to change the date and version number:

```
\ProvidesFile{rmplocal.cfg}
 [1381/04/01 v0.1 Wat Tylers's
  local configuration file for the rmpage package.]
```

It's okay to comment out and uncomment options, but don't make any other changes above the line in the configuration file that says: `LOCAL CODE BELOW HERE PLEASE`. I won't complain if you do, but you'll find it harder to upgrade to new versions of `rmpage`.

7.2 Configuration basics

I assume that you have made a copy called `rmplocal.cfg` of either `rmplocal.gfc` or `rmpgen.cfg`, and that you've changed the `\ProvidesFile` command. If not, read the start of this chapter again. I will refer to `rmplocal.cfg` as the config file in this chapter; there are other names a local configuation file might have.

Please don't change your config file above the line that says: LOCAL CODE BELOW HERE PLEASE. I won't complain if you do, but you'll find it harder to upgrade to new versions of `rmpage`.

7.2.1 Unknown option error

If you are using a copy of `rmplocal.gfc` and LaTeX complains about an unknown option, edit your config file and uncomment the option you need. The more options that are uncommented, the slower `rmpage` will work.

7.2.2 Default options

Look through your config file for the line CHANGE THE COMMAND BELOW TO MATCH YOUR LOCAL PREFERENCES. There's an `\ExecuteOptions` statement just below it. This statement sets your default options—my default settings are below. I have a special printer type, because my printer is old, tired, and out-of-spec; I usually print on A4 paper with a large inside margin so I can put my printouts in a ring binder, and I like my dates like this: 5th November 1996. You can change the argument of this statement to match your preferences: if you're an American with a LaserJet 4, replace the printer option with `lj4printer`, change the paper type to `letterpaper`, and the date to `usdate`. If most of your output goes in ring-binders, keep the `notstdmargins` option; otherwise, change it to `stdmargins` to give a conventional large outside margin.

```
%%      CHANGE THE COMMAND BELOW TO MATCH YOUR LOCAL PREFERENCES
%%      ------------------------------------------------------
%%
\ExecuteOptions{R+R-dw520printer,a4paper,notstdmargins,nicedate}
%%
%%      ------------------------------------------------------
```

You can add almost any option you like to this statement: it sets the defaults for everything you typeset with `rmpage`. Some of the options in `rmplocal` have been commented out to speed things up, so you might need to uncomment them to allow them to work. You can tell whether an option needs uncommenting from the list of all the options in chapter 5 on page 45: options that look like this: *`obscureoption`* are commented out; options that look like this: `usefuloption` are not commented out. None of the options in `rmpage` are commented out—you only need to edit the configuration file.

You can have additional default options for particular classes. Because the options for particular classes are executed after the general defaults, they can over-ride the general defaults. So it's quite all right to say `a4paper` in the `ExecuteOptions` statement above if you normally print out slides on B5 paper, because the `b5paper` option executed later on will over-ride the original `a4paper` option.

If you want different default options for different classes, or if you intend to use the slides class, read the next section which explains all.

7.3 Configuring rmpage for particular classes

Just below the `\ExecuteOptions` statement which sets the global defaults, there's a section headed DEFAULT OPTIONS FOR PARTICULAR CLASSES. This is the place intended for your class-specific default settings. `rmpage` provides a command to do this:

```
\RM@OnClassExecuteOptions{<class name>}{<comma separated options>}
```

It's just an `\ExecuteOptions` command that is only executed for the named class. For example, my configuration file has the statement:

```
\RM@OnClassExecuteOptions{slides}
  {centre,ukdate,R+R-dw520printer}
```

Which executes the given options only when I'm using the `slides` class. You might like to change this statement to match your preferences.

You can have as many of these `\RM@OnClassExecuteOptions` statements as you like, although one statement for each class is probably best. If you are creating a class of your own, say a thesis class based on `report`, it's probably best to make sure that the default options for your thesis class are executed after the default options for the `report` class. That way, the `report` class's defaults don't over-ride your thesis class's defaults.

If you want to create a new class with the help of `rmpage`, read section 7.6 on page 81 for more details.

7.4 Defining a new printer type

7.5 Dealing with particular combinations of printer and paper

7.6 Telling rmpage about a new class

If you are going to define a new class type, there are two obvious ways of doing it: either declare a new option which sets the `\RM@classtype` command to your new class number (above 100, please), or put a line in the `\RM@DefineNewClasses` hook:

```
\DeclareOption{nuthesisclass}{\def\RM@classtype{101}}
```

or

```
\newcommand*{\RM@DefineNewClasses}{
\RM@SetClassType{nuthesis}{101}
}% endRM@DefineNewClasses
```

The advantage of not using an option is greater speed. The advantage of using an option is that you can pass the option to `rmpage`, and be sure that your particular settings are acted upon, even if you change the name of your class.

You can set default options for a particular class in the config file: the DEFAULT OPTIONS FOR PARTICULAR CLASSES section is for you to add:

```
\RM@ClassExecuteOptions{<class name>}{<options list>}
```

statements for each class you want to `\ExecuteOptions` for. make sure that if you are building one class upon another (e.g., building nuthesis on report), that you execute the options for the base class first (e.g., do report before nuthesis).

The `\RM@AfterProcessOptions` hook is the ideal place to use the `\RM@OnClassType` command to set up things for particular classes. You can set things like minimum margins, maximum textwidth, and the like there. See the config file for some examples.

7.6.1 Dealing with options

If you want to build a new class by modifying a standard class with the help of `rmpage`, you need to think about what's going to happen to options.

You can tell your new class, let's call it the `nuthesis` class, to pass options on to `rmpage` quite happily, by including a line:

```
\DeclareOption*{\PassOptionsToPackage{rmpage}
    {\CurrentOption}}
```

in the option declaration section of your class file to pass all options you don't deal with to `rmpage` (not forgetting to say:

```
\ProcessOptions
...
\RequirePackage{rmpage}
```

later on).

The problem with this is that any options you have set up explicitly (for example, you might have pass the `wide` option to rmpage to get a particular `\textwidth`, and the user might have asked for `narrow`. Do *you* know which one takes precendent?) might be over-ridden by the user.

The safest way to deal with this is to decide which of `rmpage`'s options you will allow the user to use. You might stick with the

```
\DeclareOption*{\PassOptionsToPackage{rmpage}
    {\CurrentOption}}
```

statement in your class file and prepare a special `rmpage` config file for your new class, which has all other options commented out. If you call this config file `rmpage-nuthesis.cfg`, include the line

```
\newcommand*{\RMconfigfile}{rmpage-nuthesis.cfg}
```

in your class file before loading `rmpage`. Or you might use:

```
\DeclareOption{<option name>}
    {\PassOptionsToPackage{rmpage}{<option name>}}
```

to pass each allowed option on to `rmpage`, and ensure that you're not passing options to `rmpage` with `\DeclareOption*`.

If you have used `rmpage` to help you get printing dimensions right for a fixed format and you don't want the user to change anything, you might find it

best to set the various parameters directly in your class file, and forget about using `rmpage` entirely. You can find out what they were set to by looking in the `log` file; by the time I've released `rmpage`, everything that `rmpage` changes damned well ought to be noted there, and if not you can curse me for a careless fool, and specify the `garrulous` option to `rmpage` which will print everything and then some in your TEX console window. Don't try this if you're going to keep using `rmpage` to build your class—I have no idea what'll happen.

7.6.2 Things you can do with your new class number

Changing textheight setting

You can define the command `\RM@textheightgroup` to any number you like. It's probably best to do this in the `\RM@AfterProcessOptions` hook.

Currently, the initial value of textheight is set by one of two blocks of code: one is executed if `\RM@textheightgroup` is 0 (default); the other is executed if `\RM@textheightgroup` is 1 (slides only).

If you set `\RM@textheightgroup` to anything other than 0 or 1 for any of your classes, you will need to add some code to set textheight! Put your new code in the `\RM@BeforeTextheightSetting` hook; have a look at `rmpage` to see how I did it.

Changing textwidth setting

ALL THIS IS WRONG NOW!

You can also define the command `\RM@widthsetter` to be any filename you like, using any of the hooks executed before the width setting code is used. If the command is not defined just before the `\RM@BeforeWidthSetting` hook is executed, it is defined to be rmpwnorm.pko (the extension stands for package option). If the class type is 5 (`slides`), the command is then defined to be rmpwslid.pko. This filename is the file loaded to set the various horizontal parameters. By default, `rmpwnorm.pko` is loaded for all classes expect slides, which uses `rmpwslid`. An example of this sort of thing is this fragment of config file code:

```
\newcommand*{\RM@AfterProcessOptions}{
    \RM@OnClassType{101}{% class 101 = nightmare university thesis class
        % Use different textheight setting code to everything else.
        \def\RM@textheightgroup{2}
        % Set minimum margins as specified in the regulations.
        % Everything else is done by the class file.  These commands
        % are defined to be 1742pt at the start of rmpage, so they
        % can't be set in the class file.
        \def\RM@minoutsidemargin{15mm}
        \def\RM@mininsidemargin{40mm}
        \def\RM@mintopmargin{15mm}
        \def\RM@minbottommargin{15mm}
        % set minimum and maximum textwidth, as defined by the regs
        \def\RM@mintextwidth{130mm}
        \def\RM@maxtextwidth{160mm}
        % load custom width setting code in file vulture-widths.jkl
        \def\RM@widthsetter{vulture-widths.jkl}
    }{}
}
```

```
%
%
%
\newcommand*{\RM@DefineNewClasses}{
    \RM@SetClassType{rmcv}{20}
    \RM@SetClassType{rmletter}{21}
    \RM@SetClassType{bithesis}{22}
    \RM@SetClassType{ljmueepexam}{23}
    \RM@SetClassType{nuthesis}{101}
}% endRM@DefineNewClasses
%
%
% nightmare u. thesis uses fixed total text area height of 7in
\newcommand*{\RM@BeforeTextheightSetting}{
    \RM@OnTextheightGroup{2}{%
        \setlength\RM@totalheadfootclearance{\paperheight}
        \addtolength\RM@totalheadfootclearance{-7in}
    }
}% endRM@BeforeTextheightSetting
%
```

The code fragment above defines a new class type, `nuthesis`, number 101, which is a class for preparing theses for Nightmare University. This class uses textheight setting code that asks for a total text body height of as near to 7 in as possible no matter what textheight options are specified, and horizontal text parameters are set by the file called `vulture-widths.jkl`. The other limits on the printing region specified by the university's regulations are placed in the `\RM@AfterProcessOptions` hook. What I have typed above is in addition to any code which might be in those hooks anyway; don't remove anything unless you've got a good reason—who knows what might go wrong?