A Writing System
Version 1.2

Richard Baron

18 November 2015

Version 1.2, published November 2015

Contents

1. Introduction

This is a system for writing substantial texts. We cover both keeping work organized and the checking of work.

A .tex file and a PDF file are available at
http://www.rbphilo.com/authors.html

Everyone is welcome to copy the files and adapt the system to their requirements. See page ii for the applicable licence.

There is a general presumption that you will use LaTeX, but most of what is said can be applied with only minor changes when other systems are used.

The system also reflects use of the Git version control system. If you do not use Git, you should ignore paragraphs that start with "**(G)**". When "**(G)**" occurs within a paragraph, you should ignore the remainder of the paragraph.

The system has been set out on the basis that there will be one author, working on one computer or one set of files in the cloud, to prepare a document. (We shall use the term "document" to refer to the complete work that is to be produced.) A system in which Git or something similar played a central role would be needed if there were several authors, or if several versions of a document existed at once.

2. Software

2.1 Producing documents

You will need to choose a single package for writing material. LaTeX is strongly recommended, for several reasons. You can compile a PDF to read with one click. You can comment out a paragraph or a command line, so that the paragraph does not appear in a compiled PDF or the command line has no effect, simply by inserting a percent sign at the start, and you can reverse the effect simply by deleting the percent sign ("uncommenting"). Compiled PDFs look beautiful. And since .tex files are text files, they can be read on any computer, compared with one another, and used with version control systems like Git.

You will need a file comparison program. Any package that will allow comparison between two files of the types produced by your chosen writing package will do. The program should display the two files side by side, and highlight differences between them.

You may wish to use a package to put in references and compile a bibliography automatically, for example BibLaTeX. (The alternative is to enter all the references and compile the bibliography by hand, which can be tedious.)

It is important to choose the package at the start and to stick with it, so as to avoid any need for changes to your bibliography file. But before making a choice, you should experiment with the package you have in mind to ensure that it will produce inline references, footnote references and bibliography entries in the styles that you will want for different items (books, book chapters, papers, conference proceedings, and so on).

2.2 Backups

Remote automatic backup of files is a wise precaution, even if you also plan to save work to an external hard disk or a memory stick every day. Any backup system must be set to copy hidden files as well as non-hidden files if you use anything that generates hidden files (Git does). And frequent backups to external disks or sticks must still be made, always keeping all copies of files and not allowing new versions to overwrite old ones, just in case the remote backup service goes wrong.

(G) Git is a powerful version control system. Its full power is revealed when you want to create branch versions of a piece of work or you want to work with others on the same document. We shall not cover those possibilities here. But even if you are a single author working on a single version of a document, Git gives you an extra way to trace the changes you have made to files, and (if you use a service like GitHub or GitLab) an extra backup.

3. Folders and Files

In this section we recommend a structure of folders, and say a bit about the files that should go in each folder. Later sections will cover procedures which will lead to the folders gradually being filled up with files. We shall put folder names in **bold**.

You can of course vary the naming scheme, but it is important to choose your naming scheme at the start and then stick to it.

3.1 A structure of folders

3.1.1 Outline of the structure

writing

- **projectname**
  - **chapters**
  - **main**
  - **management**
  - **notes**
- **standing**
  - **bibliography**
  - **checklists**
  - **latexstyles**
  - **styleguide**

wxarchive

- **wxprojectname**
  - **wxcompare2**
  - **wxmaterial**
  - **wxoldversions**
- **wxstanding**
  - **wxcompare2**
  - **wxoldversions**

3.1.2 The contents of the folders

- **writing**. **(G)** This whole folder will be a single Git repository if you use Git – but see the note in section 3.1.3.
    - **projectname**. There will be one of these for each project. If there is any risk of accidentally re-using names, start the project name with the year and month of commencement, for example 1512 for December 2015.
        * **chapters**. This folder will contain the up-to-date files of chapters intended to appear in the document you are writing. Files of preliminary material and any bibliography to be compiled by hand (rather than automatically) are treated as chapters for this purpose.
        * **main**. This folder will contain the main LaTeX file that will pull in the chapters, the relevant .sty file and, if your bibliography is to be compiled automatically, output from the bibliography file. Files will be in other folders, so you will need to specify paths to them. **(G)** When you compile PDFs, a lot of extra files will get generated, so you will need to set up a .gitignore file to make Git ignore files of the relevant types (.aux, .log, and so on).
        * **management**. This folder will contain a file that will control the flow of work. It will also contain checklists.

    - **notes**. This folder will contain files of notes you make as you work. However, notes that relate to particular points in chapters should generally be made in those chapters, and commented out so that they do not show up in compiled PDFs.

- **standing**. This is a single folder for all projects.

    - **bibliography**. This folder will contain a single bibliography file. You will copy the entries you require to an appropriate file in the relevant **chapters** folder near the end of work on a project, or alternatively use the bibliography file to generate each project's bibliography automatically.

    - **checklists**. This folder will contain master copies of all the checklists you use. It will also contain a master copy of this writing system, adapted to your requirements.

    - **latexstyles**. This folder will contain all the LaTeX (or other system) style files that you might want your software to use when preparing documents. This folder is for instructions to software, not for a style file in the sense of instructions to yourself on preferred spellings and the like.

    - **styleguide**. This folder will contain a single file that will set out the rules you wish to follow on spelling, punctuation, italicization and the like.

- **wxarchive**. "wx" is included to make the folder come out just after the **writing** folder when a file manager arranges folders alphabetically. "wx" is also included in the names of the subfolders, to avoid confusion with folders in **writing**. The **wxarchive** folder should be covered by your remote automatic backup. **(G)** There is no need for this folder to be a Git repository.
  - **wxprojectname**. There will be one folder like this for each project.
    - **wxcompare2**. This folder will contain copies of two recent versions of files from **projectname**. It is only a temporary staging post. The name of the folder includes a reminder that it should only ever contain two versions of any given file.
    - **wxmaterial**. This folder will contain source material obtained from elsewhere, as distinct from your own notes. You can set up any sub-folders you may find useful within this folder.
    - **wxoldversions**. This folder will contain earlier versions of files from **projectname**. You can set up any sub-folders you may find useful within this folder.
  - **wxstanding**. This folder will contain earlier versions of files from **standing**. The sub-folders **wxcompare2** and **wxoldversions** should be used in the same ways as the corresponding sub-folders in **wxprojectname**.

3.1.3 The number of Git repositories

(G) We said above that the whole of the **writing** folder should be a single Git repository. This reduces the amount of work that needs to be done to commit all changes, and therefore the amount of work needed to use a remote repository as an extra backup. But if you use Git's branching function much, or if several people work on the same project, you may prefer a separate Git repository for each project, and an additional repository for **standing**, still all within the **writing** folder. That would be a perfectly good way to work, but what is said in section 4.2 about adding files to Git and about staging, committing and pushing changes would need to be modified accordingly.

3.2 Files

3.2.1 The document

Files of text for your document will all go in the **chapters** folder. If you use LaTeX, you should also have a main file, stored in the **main** folder. Individual chapter files will be pulled into it with the input command. This will allow you to keep individual chapters reasonably short, reducing the demands made on your computer. Having said that, it can be useful to be able to search the whole text for a given word or phrase. It is worth finding an editor that will allow you not only to keep several chapter files open at once, but also to search them all at once.

It is not a good idea to include chapter numbers in the names of files of individual chapters. You might re-order the chapters, and then the numbering would be confusing.

You could of course change the numbers in the names of files at that point, but then you would have to take care to make corresponding changes to commands in the main file. It is simpler to do any re-ordering purely by re-arranging lines in the main file, and to consult that file if you need to be reminded of the order in which chapters come.

3.2.2 Management

management will contain a single workflow file for the whole project, in which you will enter tasks to perform, in the order in which you plan to perform them. The file will naturally change as you go along. Completed tasks should not be deleted. Instead, brief notes on what was done and when it was done should be added, and the relevant lines should then be commented out so it is immediately obvious on screen that the tasks have been completed.

It is essential to have only one workflow file, so that tasks do not get overlooked because they are in a different file. The workflow file may refer to checklists and to other files that give detailed descriptions of tasks, but all of them must be subservient to the workflow file, and every task must appear in that file.

Checklists in use or completed will also be stored in **management**. We shall cover this topic in section 6.1.

3.2.3 Standing files

3.2.3.1 Bibliography

A complete bibliography file will go in **bibliography**, in the format that you wish to use (a .tex file of entries to be copied and pasted as required, a file for BibLaTeX to use, or whatever). As you work, you will add new entries to this file, not directly to the bibliography of the current project.

3.2.3.2 Checklists

You will want to devise checklists for various purposes, and master copies of them should be stored in **checklists**. When they are to be used you will make copies, as we shall explain in section 6.1.

Your personal style guide will also serve as a checklist, but it should have its own folder, and not be reproduced here. You should not have multiple copies of any master document, because amendments might not get made on all copies.

3.2.3.3 LaTeX styles

If you use separate LaTeX .sty files, rather than incorporating styling information directly into documents being produced, those .sty files should go in **latexstyles**. Your main file for each project can then call the appropriate .sty file, using the usepackage command.

You can of course amend a .sty file as you work, but all amendments should be made to the appropriate file in **latexstyles**, not by adding extra commands in documents.

This may mean that you will sometimes create a new version of a .sty file, with a new name, so you can make changes that you do not want to apply to all documents that use the existing .sty file.

3.2.3.4 Style guide

styleguide will contain a single file, your personal style guide. If you use a published style guide, your personal guide will only need to cover points that the published guide does not cover.

Examples of points of style you may wish to cover include preferred spellings (such as forward or forwards), what you capitalize, what you italicize, how many authors you will list when giving a reference and how many will lead you to use “et al.”, and whether you will write “chapters 2 to 4” or “chapters 2-4”.

You will add to this file as you go along and new points occur to you. Text already written may not comply with your new decisions, but that will not matter because the checking stage at the end will include a check for compliance with your style guide. We shall cover this in section 6.3.6.

3.2.4 Wxarchive files

wxmaterial is to be used for source material obtained from elsewhere. We shall not suggest how it should be used, except to say that it should be thoroughly backed up (a topic we shall cover in section 4.2).

We shall explain the files that will go in **wxcompare2** and **wxoldversions** in section 5.

4. Writing

We shall speak of writing sessions (this section) and comparison sessions (section 5). A writing session is simply a stretch of time, usually fitting within a single day, over which you will write text for your document, for notes files, for management files or for standing files, or gather materials. It may include making amendments to what you have already written. A comparison session will involve comparing a current version of some text with an earlier version and making amendments.

4.1 During each writing session

Writing can take place anywhere within **writing**, and source material can be collected in **wxmaterial**. **(G)** If you use an external Git repository and you might have changed anything in it other than by making changes to files on the computer you are currently using and then pushing to that repository, you should start by pulling from the external repository.

It is a good idea to save files every few minutes, so that your remote automatic backup service takes copies. You may wish to set your editor to save automatically. Having

said that, there is a risk associated with automatic saving (and with manual saving if the action becomes practically automatic). If you accidentally delete a chunk of text you wrote in the current session and the file is then saved, you could lose the deleted text. (If the deleted text was written in a previous session, you will have a backup.) If this happens, it is important to stop immediately and retrieve a previous version from your remote backup service before it falls off the list of available versions.

You will need to decide how to mark queries, for example notes that certain facts need to be checked, that a paper you dimly recall needs to be found, or that you need to decide between alternative ways to state a point. It is not wise to comment out queries, because then they will not show up in PDF files that you compile for review. We suggest marking them with five asterisks in a row, *****. This should make them very conspicuous. The reason for using five asterisks is that if you wish to search a PDF for all instances you can search for two asterisks, **, and be confident that every instance will be found even if some of them are hyphenated over two lines. Some PDF search facilities do not detect strings that are split over two lines, but if you use five asterisks, there will be at least two together on a single line. (Three asterisks would in theory suffice, but if you use five, you will still have enough even if you sometimes omit an asterisk or two by mistake.)

References, quotations and bibliography entries can be checked as you work. They should not be checked straight after writing them, because then you will see what you expect to see. They should be checked after a few hours or, better, the next day. You can mark points to be checked with something like “ ***** Check Tuesday” to give you a quick way to locate points to check.

Notes on checking done should be made in the relevant files, as and when checks are made. These notes should be detailed, for example "Checked the words 'This theory is absurd' and page number 37 to printed copy published in 2004 in the British Library on 28 January 2016". If a note relates to something in a chapter, it can be placed at the end of the relevant paragraph in the chapter and commented out, so that it can be kept in permanently without ever showing up on a compiled PDF. (A note should not be placed in a fresh paragraph, because then it might become detached from the relevant paragraph if new paragraphs are inserted.) If a note refers to an entry in a bibliography that is simply a .tex file, the same can be done. If you use a system like BibLaTeX, a note on checking can be placed in a note field or some other field that will not be used to generate text to appear in references.

The library catalogue numbers of books you consult can also be noted in the same way.

4.1.1 Compiling PDFs

At various stages when writing or checking, you may wish to compile a PDF so as to see the state of your document. You will do this in the **main** folder. If you only want to compile some chapters, simply go into the main file and comment out the lines that pull in the chapters you do not want. You can uncomment these lines later.

When you use PDFs to see what you have written, it can be helpful to have the lines spaced out. One option in LaTeX is to use the openup command. If you include this in your main file you can get wider linespacing every time you compile. When you want to prepare a final version, you simply comment out this line.

4.2 Backup steps

Each writing session should end with the following backup steps, in this order.

1. Ensure that any new files have the names you will want them to keep. For example, if you have created a checklist to use by copying a file from the **checklists** folder to the **management** folder, you will want to add the date you started to work through the checklist to the name of the copy in **management**.

2. **(G)** If you are using Git, add any new files within the Git repository to Git so that they get tracked. (This must be done after names are assigned, because Git can go wrong if files change their names.)

3. **(G)** If you are using Git, stage and commit all changes, and then push to your external repository if you use one.

4. Plug in your external backup (we shall call this a memory stick for convenience, but it could be an external hard drive). Create a new folder on the memory stick and give this new folder a name that includes the date (and the time if you might create more than one such folder on the same day).

5. Copy the whole of your **writing** folder to within this new folder. The first time you do this, or use a new stick or a new file manager, you should check that all hidden files get copied over. The point of this extra layer of folders on the memory stick is to avoid any changes to folder names. Changes of name may lead to confusion. **(G)** In addition, they can upset Git.

6. Back up all new or changed items in your **wxarchive** folder to the memory stick. The first time you use a stick, you will simply copy the whole folder. After that, you can limit yourself to copying files that are new or have changed. But given that storage is so cheap, you may prefer to copy the whole folder every time. If you do so, you can use the dated folder to which you copy the **writing** folder.

5. Comparison

Every so often you will want to go back over text you have written and compare it with an earlier version, so that you can reconsider all the changes you have made. This is most likely to be useful when you think that a chapter is more or less complete, and you are polishing it or checking details. But you can follow the procedure described in this section at any time. And it can be used for anything in the **writing** folder, including management files, notes files and standing files.

This procedure only covers reviews that involve comparison with an earlier version. Changing text as you go along counts as ordinary writing, and only the procedure set out in section 4 need be followed.

Each comparison session must be immediately preceded by the full backup procedure described in section 4.2. So if you have a writing session and then a comparison session, you should treat the two as separate sessions.

5.1 Initial set-up

When you first have enough text that you think you might want to use it as the starting point for comparison with a later version, go through the following steps.

1. Copy the file in question to the **wxcompare2** folder (in **wxprojectname** for anything from a **projectname** folder, in **wxstanding** for anything from the **standing** folder).

2. Change the name of the copy in that folder by appending the date you made the copy in a form that will make sorting by name give the same order as sorting by date (for example, 160327 for 27 March 2016). Add the time to the name if there is any chance you may make two copies of the file on the same day.

3. Make the copy read-only. This is to prevent your accidentally doing further work on what is only a copy.

5.2 The comparison procedure

When you want to compare a file with a previous version, start by following the steps in section 5.1 for the latest version of the file – including making the copy read-only.

You will now have two copies of the file in **wxcompare2**. There may be copies of several files in the folder, but that will not matter. The two copies of a given file will appear next to each other in a listing of files by name. You can then go through the following steps.

1. Compare the two copies in **wxcompare2**, using your chosen file comparison program. As and when you come to differences, consider whether the changes you made were improvements. While doing this, you may find it helpful to look at a PDF compilation of the latest version, as described in section 4.1.1.

2. Make any further changes to the original of the file, in the **writing** folder.

3. After making any changes, do not copy the latest version to **wxcompare2** unless you first complete the whole procedure described in this section and then repeat it from the beginning. There is in any case no need to copy the latest version, because the currently newer file in **wxcompare2** will become the older file for the next comparison, so changes you make this time round will be presented for review then, along with any subsequent changes.

4. When you have completed your review, move the older of the two versions in **wxcompare2** to **wxoldversions**. The newer version stays in **wxcompare2** to await the next comparison session.

5. Finish the session by going through the entire backup procedure that we set out in section 4.2. Do not attempt to take any shortcuts on the basis that you went through this procedure at the start of the comparison session.

It may be asked why we propose a **wxcompare2** folder, rather than simply copying files directly to **wxoldversions** and making the comparison between the most recent two versions of a file in that folder. We propose the **wxcompare2** folder because it gives an additional control to make sure that all changes are reviewed – something

that is especially important in the closing stages of a project, when changes are made as a result of checking and those changes themselves need to be checked. If you do not allow yourself to move a file from **wxcompare2** to **wxoldversions** until it has played the role of the older file in a comparison, and you also make sure that there are never more than two versions of any one file in **wxcompare2**, every change will be reviewed. Having this arrangement also allows you to copy a file directly from **writing** to **wxoldversions** at any time should you wish to do so, without fear that changes will fail to be reviewed. (If you do so, you should of course append the date to the copy's filename and make the copy read-only.)

(G) The same concern about the risk of not reviewing changes is an argument against use of the facilities of Git to review changes. If all changes are to be reviewed, it is essential to compare the current version of a file with the version that was last used for a comparison, and that may not be the version at the time of the most recent commit. So if you do want to use Git rather than the procedure set out here to review changes, you must keep a careful note of which commits to use. You could of course use Git commit messages to state whether files as at the time of a particular commit were about to be used for comparison with earlier files, so that they could themselves be used as the earlier files in later comparisons, but this would require care if commits were made for several files at once, because some would have changed while others had not. You might accidentally fail to compare versions of a file that had changed, and then go on to use a changed version of the file as the earlier file in a subsequent comparison.

6. Checking

Once your document is complete and is in a reasonable state, you will enter the checking phase. This has three stages: polishing the document, checking details, and checking the typesetting.

The first and second stages are likely to overlap. When you check details, you may well notice additional ways to improve the document. But both of these stages should be completed before you move on to check the typesetting.

You will probably need to perform stages more than once. A first polishing and checking of details will detect lots of errors. A repetition will detect far fewer, but it may well detect errors that were missed at first because of the quantity of errors.

We shall first make some points about stages of checking in general, and then say something about the types of check to make at each stage.

6.1 Stages of checking in general

Checking will involve both reading through the document and working through checklists.

You may wish to work on a printed copy, in which case you will need to tick off handwritten amendments when you transfer them to files in the **writing** folder.

Checking is most likely to be effective if you use a compiled PDF with increased space between lines, as described in section 4.1.1. This is so whether you work on screen or on a printed copy.

When a checklist is to be used, the first step is to copy the relevant file from the **checklists** folder to the **management** folder, then add the date you start to work through the checklist to the name of the copy in **management**.

As you work through a checklist, do not delete lines in it. Add a note to each line that states when you completed the task and then comment out the line. Completed lists will then collectively provide a detailed diary of work done.

An outline diary should also be kept in the workflow file for the project. Entries will include references to the dated checklists by name, so that you can go to those checklists and see exactly what you did at each stage. You can also use the workflow file to record exactly where you have got to in a task that stretches over several sessions.

Throughout the process of checking, you should continue to follow the procedures of writing and comparison set out in sections 4 and 5. It is important to ensure that every change gets reconsidered in a comparison session, so you can

be confident that sessions of checking will not undermine previous checking because you can expect comparison to find any changes that you make by accident. No stage of checking should be regarded as complete until there has been a comparison session which has shown that no changes have been made since the previous comparison.

6.2 Polishing

Polishing is simply a matter of reading and revising your words.

A useful technique is to read out loud – in a normal speaking voice, not in a whisper. This will draw your attention to inelegant turns of phrase more effectively than silent reading. Reading out loud can also help when checking details.

It is useful both to spend time on small sections, especially when there have been many changes to them, and to spend time toward the end of the checking process reading through the whole work, or at least whole chapters, continuously. Such extended reading is important to make sure that the whole work hangs together even after the many changes that will have been made.

6.3 Checking details

We here list several points that will need to be checked, and re-checked if there are changes to the document. Such re-checking can be tedious. One option is to compile a checklist for all of these points and others that may be

relevant to your work, to apply a copy of that checklist to the whole document when you think the document is nearly final, and then to work through fresh copies of the checklist but only to cover text that has been changed (changes will be disclosed in comparison sessions). Having said that, you may judge that some points should be checked again throughout the document.

6.3.1 Preliminary pages

Check that everything in these pages is accurate.

Check that all acknowledgements of help and of permission to reproduce material have been made.

Check that any necessary information on how to use the document, on versions, or on how references have been given is included.

6.3.2 Summary

If the document includes a summary, check what it says to the main text.

Chapter and section headings repeated in the summary should also be checked.

6.3.3 References and quotations

Check that all references and quotations have notes that state when and how they were checked for accuracy, as described in section 4.1. Take particular care when any

reliance has been placed on electronic texts that may not have exactly the same pagination as printed versions.

Ensure consistent application of your policy on quoting titles or text in languages other than the main language of your document, and on supplying translations.

Check that all references are in the desired style (details given, italics, quotation marks, punctuation and the like).

Check that all references have corresponding entries in the bibliography. Even if references and the bibliography are generated automatically, a check may be worthwhile.

Ensure that all necessary permissions for quotations from in-copyright works have been obtained. If in doubt, replace quotations with paraphrases.

6.3.4 Bibliography

Check that all entries in your bibliography file have notes that state when and how they were checked for accuracy, as described in section 4.1. Take particular care when any reliance has been placed on electronic texts that may have different bibliographical information from printed versions.

Check alphabetical order in the bibliography generated for your document, making sure that it complies with any rules you wish to adopt, for example on the treatment of “Mac” and “Mc” or on the treatment of letters with diacritics.

Check that all entries you want to appear in the bibliography for your document, do appear. For example, if there is an entry for a paper in a volume, you may want an entry for the whole volume as well (although you will probably not

want that if the volume is a book of papers exclusively by the author of the paper). To take another example, if there is an entry for an item in a language other than the main language of your document and a translation is available, you may also want an entry for the translation.

Check that all entries in the bibliography for your document are used in the body of the document.

Check that all bibliography entries are in the desired style (details given, italics, quotation marks, punctuation and the like).

6.3.5 Internal cross-references

Check that all internal cross-references point to the right places and that they work as links (assuming that you have set up your document to make them clickable links). They can be found in a .tex file by searching for the ref command, and tested in a compiled PDF.

If you are using LaTeX, search a PDF for "??", which is what will appear if a cross-reference uses a label that is mistyped or is not attached to any location in the document.

6.3.6 Style

Copy your personal style guide to **management** and put the date into the copy's filename, then use the copy as a checklist to work through your document and check for compliance with your guide. Also check everything to any published style guide that you have decided to use.

When searching for instances of non-compliance, such as spellings you do not wish to use, it is better to search .tex files than to search a PDF, because searches of PDFs may not find words that are hyphenated over two lines.

It is worth thinking of quick ways to find instances of non-compliance. For example, if you want to use “forward”, “backward” and so on rather than “forwards”, “backwards” and so on, you can search for “wards”.

Changes to ensure the use of preferred spellings should not be made by globally exchanging some strings for others, because then you might accidentally change spellings in titles or quotations. You might also accidentally change parts of words.

6.3.7 Brackets

Check that left and right hand brackets all match up. As well as scanning the document, it makes sense to count numbers of left hand and right hand brackets of each type (round, square and curly) automatically, to ensure that for each type, there are as many left hand ones as right hand ones.

If your editor does not provide such counts, make copies of the relevant .tex files somewhere outside **writing** and **wxarchive**. Then within each copy file make global exchanges of each type of bracket (left round, right round, and so on) for some other string (say xxxx), note the number of exchanges made each time (likely to be shown at the bottom of the screen), and check that the numbers match. At the end, delete the copy files you have used for this exercise.

6.3.8 Links

In a PDF, click on web addresses that you want to work as links to make sure they work and point to the correct pages. You can find links by searching your .tex files for "www" and for "http".

In a PDF, click on all links in your bibliography to make sure they work and point to the correct pages. Bear in mind that even DOIs can cease to work, for example when journals are moved to new publishers and the DOIs for old papers are not redirected to the new publishers' archives.

6.3.9 Remaining queries

Search for instances of the mark you have used for queries and clear any remaining queries. In section 4.1 we recommended marking queries with *****, so you could then search for **, even on a PDF, and find all instances. But it is still safest to search your .tex files as well once you think you have cleared all queries.

Search your .tex files for any strings you may have used to mark places you reached while working, such as "xxx", "yyy" or "zzz".

6.3.10 Spelling

Spelling will need to be checked more than once (starting afresh each time), making the final check very near the end of the whole process of checking.

Spellcheckers are essential, but they can also be dangerous. It is all too easy to click “ignore” or “ignore all” automatically, or to accept the spellchecker’s suggestion when you do not really want to do so.

One option is to make copies of your .tex files somewhere outside **writing** and **wxarchive**, use a spellchecker on the copies, and every time it queries a word, substitute xxxxx for the word (unless you are absolutely certain that it is something a spellchecker is bound to query mistakenly, like the name of an author). Then you can go back over the copy files, find each instance of xxx (search for three x’s even though you inserted five, in case you accidentally inserted too few), find the corresponding place in the appropriate file in **chapters** or in **bibliography**, and make changes there.

Corrections should not be made by globally exchanging some strings for others, because then you might accidentally change spellings in titles or quotations. You might also accidentally change parts of words.

6.3.11 Inappropriate characters

If you use LaTeX, you should search .tex files for curved apostrophes, curved single and double quotation marks, and straight single and double quotation marks (find straight single marks by searching for space followed by straight apostrophe, so as not to find every apostrophe, as well as looking out for single marks next to double marks). Replace such characters with straight apostrophes (for apostrophes and closing quotation marks, used in pairs for double quotation marks), and with grave accents (for opening quotation marks, in pairs for double marks). LaTeX may cope perfectly well with curved apostrophes and quotation marks, but the old-fashioned way is safer.

Likewise you should search for characters with diacritics which it may be wise to replace with spelt-out LaTeX commands (such as the apostrophe to get an acute accent).

You should search for single hyphens where you might have wanted to use double hyphens to create dashes, and to check whether you have made dashes bold by using the textbf command if that is what you wish to do. (It does make them show up better.)

If you are using something other than LaTeX, you should search for instances of two spaces in a row, and eliminate unwanted spaces. (LaTeX should ignore extra spaces.)

6.4 Checking typesetting

Once you are ready to compile a PDF with the linespacing you want, rather than the wide spacing you may have introduced in order to facilitate checking, you will need to do so and then go through the PDF to check its appearance.

Changes to improve typesetting naturally have an effect on what follows them, so it is important to work forward through any document, and to recompile it after making each change.

There does need to be a final comparison session for each .tex file after making all changes, so as to ensure that the only changes made are ones made for typesetting reasons and that they are all acceptable. There should also be a final review of a PDF of the complete document.

Questions to ask when checking typesetting include the following.

6.4.1 Pagination

Do you have an even number of front matter pages and an even number of main matter pages (unless odd numbers will be acceptable given the ways in which your document will be used)?

Do you have a total number of pages that will meet any printer's requirement (for example, a multiple of 16)? This may however be something that the printer will sort out by inserting blank pages at the end, without your needing to do anything.

Are all pages that should be odd-numbered in fact odd-numbered (for example the first pages of the table of contents, of the preface, of each chapter, of the bibliography and of the index, unless you have chosen to allow chapters and so on to start on even-numbered pages)?

6.4.2 Headers and footers

Are all running heads correct? (One danger is this. If you use chapter titles on left hand pages and section titles on right hand pages, and then do not get into a numbered section until a few pages into a chapter, right hand pages at the start of the chapter may lack running heads.)

Are all page numbers correct?

Are headers and footers correctly used or eliminated on blank pages, on the title page and on the imprint page?

6.4.3 Page layout

Is the vertical spacing between paragraphs and above and below headings correct?

Does any text stick out into a margin or fall short of a margin?

Are there unacceptable solitary short lines at the beginnings or ends of pages (widows and orphans)?

Does each bibliography entry fall on a single page?

6.4.4 Hyphenation

Is the hyphenation in headings, in the body of the document and in the bibliography acceptable?

Have any extra hyphens appeared in links? These should be avoided, because people may think they should appear as parts of the links. In LaTeX, they can be eliminated by using the href command and inserting linebreaks within the text of the link that appears in the compiled document (not in the text that generates the link). But links must be re-tested to make sure they work and point to the correct pages after any changes like this are made.

6.4.5 Searchability, fonts and margins

Is the compiled PDF searchable?

If the PDF is to be printed, do the fonts used need to be embedded? If they do, make sure that they are.

If the PDF is to be printed, has a version with appropriate margins been produced? Equal left and right margins are best for versions to be read on screen, but large inner and small outer margins are generally needed for printing.

Are there any other technical requirements for printing?