WORKING NOTE AC118

Alan Creak
5 December 1997

MAKING THE HTML VERSION OF THE HANDBOOKS

These are some notes recording bits of what I did to construct the HTML files which present the department's handbooks for 1998. All being well, it won't happen this way again, but there are some points which will be significant for the eventual long-term solution, however it works.

The 1998 handbooks were perhaps the first to be converted into an HTML version with a view to setting up a permanent and more or less automatic mechanism to do the job. This is an account of what I found was involved in the process. The method I used was very much an expedient, and not intended to be repeated, but in the process I have, of necessity, found out what has to be done, and something of how it must be done. Because of the limited life of the method, though, I have not tried to produce an efficient method, so the details might not show up well under critical examination. But it works.

My major tool throughout the conversion was Word 6 for Macintosh. The reasons for using Word 6 were:

- first, it was there[1];
- second, I am accustomed to using Word;
- third, all the data were in Word-readable files;
- fourth, macros can be used in Word 6 (otherwise I'd have preferred Word 5, which is much better).

It will be observed that none of these is a matter of deep philosophical (or other) principle. Even the macros are for convenience rather than essential machinery; without them, I'd have had to work through sequences of instructions (see below) laboriously, and undoubtedly inaccurately, by hand and frequently.

The conversion was accomplished in several stages, probably because I was feeling my way. I can see no sound reason why it should not all happen as a simple sequence of operations once it has been codified, though there are certainly good reasons for being able to operate individual components independently so that amendments made during the year can be incorporated in the HTML files without reconstructing the whole handbook system.

There are sound reasons why different parts of the handbook should be converted in different ways, for there are several areas with different sorts of structure and they are kept in different files. These cases are discussed separately below, though there are some general considerations which apply over a wider range.

REQUIREMENTS.

1 : To construct HTML files which present the material of the department's undergraduate and graduate handbooks through the world-wide web in a manner similar to (though not necessarily absolutely identical with) the handbook presentation.

2 : To add useful links within the document, or to other www material, where this would be useful and informative.

3 : To make it easy to incorporate additions and amendments where this seems desirable, preferably by changing the mark-up files and "recompiling".

PRELIMINARIES.

The raw material was the collection of information in the mark-up files described earlier[2]. To be more specific, it was the mark-up files after amendments made to the cooked files used for the printed handbooks had been copied back into them.

Ultimately, this proviso should be unimportant, but in this experimental year it wasn't; the conversion back from the handbook text was also experimental, and imperfect, so some of the markers assumed by this stage of the conversion hadn't been inserted. With a better system[3] the integrity of the mark-up file should be guaranteed, and the rather surprising results sometimes seen in the results of this attempt should not occur. I didn't try to mend the conversion macro to cope with these omissions, as it wasn't worth the trouble for a single conversion under conditions that wouldn't recur. The interesting conclusions came from investigating the treatment required to convert the normal mark-up file.

For routine use, handbook changes will be made at the mark-up level, and with an automatic system the conversion should be comparatively fast. It should therefore be simple to keep the HTML files up to date and consistent as changes are made to the mark-up files.

GENERAL NOTES.

Handbooks come in parts : The exercise of converting the handbooks into HTML made it clear that there are different sorts of handbook information, requiring different sorts of treatment. In the undergraduate handbook, there are three distinct sections :

- A preliminary part, including material special to the printed versions like title pages, and non-specific information such as dates for the year. Some of this (title pages) need never appear except in the printed version and should be maintained as simple Word files; other parts are available elsewhere (dates for the year come from registry WWW pages), or might be better be presented as auxiliary information (department directory).

- An ordinary part, including the meat of the information, and in the main consisting of fairly straightforward text. Such structures as there are are restricted to chapter and section heading, with the occasional bulleted list. Tables and HTML definition lists might occasionally be useful, but the first run through managed quite well without. The most complicated part is perhaps in dealing with diagrams imported from (always, as it happens) Macdraw; these have to be converted into GIF files, and so far I've done it by hand.

- An extraordinary part, comparatively small, but with much more varied structure such as tables. This includes the timetable, lists of allocations of credit for coursework, addresses, and other similar material. This material certainly requires more manipulation to produce satisfactorily presented HTML, and so far I've done this by hand too.

None of these sections is the list of course descriptions; that comes from the Courses file, and, though varied in structure is very regular and easy to deal with.

In the undergraduate handbook, the three parts appear in that order; in the graduate handbook, the same sorts of material appear, though the odd bits are closer to the beginning than the end. The graduate handbook also makes more reference to other data files, such as the People file, but those references (like those to the Courses file) are generally very regular and pose no special problems.

In this note I describe only the ordinary part of the handbook, and the Courses file. That leaves out all the tricky bits, but I suspect that it will cover at least the germs of most of the techniques which I shall require for the complete job. The Courses file is complicated but regular; it gives examples of dealing with many different, but quite localised, structures. The ordinary part of the handbook, in contrast, is much less regular, so that structure which is required must be specifically requested by some sort of notation in the text, and cannot be inferred from context. While more demanding formatting features might require more elaborate treatment in detail, it seems likely that the patterns found in these two examples will be a sound basis for foreseeable developments.

Inserting HTML links automatically : While there is (will be, anyway) provision for specifying HTML links in the handbook text, there are some cases where this is impossible, because the authors don't know the file names, and others where standard substitutions can easily be made. The most obvious examples of the first sort are internal links to other parts of the handbook, while standard substitutions make sense for links to the pages for the department, school, and university. Others might become evident as time passes. It would in some cases be possible to lay down rules for constructing the required names, but such procedures are not very reliable, and a fully automatic method is likely to be better.

In some cases at least, the first is easy. Links to course descriptions are very easy to manage : all occurrences of 415.nnn, with or without any of the standard suffixes, can be identified and an appropriate link added. Some links to sections within the handbook are also useful; it isn't yet clear how best to arrange that. The HTML strategy of planting a label at the required target point isn't straightforward, as the sections might be managed by different people; for the same reason, anything depending on quoting any local text isn't reliable, and as section numbers will eventually be added automatically they won't do either.

The second sort of link is not quite as easy to manage as the courses, but experiments on the 1998 files showed that in practice only a very few forms of the names were used – so standardising on a few canonical forms for the names and converting those automatically would be sufficient. Anyone wishing to use something else can always add the link by hand. An alternative is to provide standard macros for the more commonly used names which people could use if they so desired.

It isn't quite so obvious when the links should be inserted. At the moment, inserting when the mark-up files are converted into HTML seems best, but there is something to be said for having the links visible in the mark-up files.

File structure : To make the links discussed in the previous section, it is necessary to know the names of the various HTML files, and this depends on the structure imposed on the file system. Other things being equal, the simpler the structure the easier it is to construct systematic file names automatically. I inherited a structure in which the two handbooks were separated, while the files for the different parts of the handbooks were kept in their own subdirectories, and courses were similarly classified according to their stages.

This made it less easy than necessary to construct file names automatically. For example, to construct a reference to the file course415.340 it was necessary to isolate the "3" after the point, and from that to construct stage_3/course415.340. There's nothing particularly difficult about that, but there's nothing particularly easy either, and it's unnecessarily complicated : the stage-3-ness of the course is used in two places, which is redundant.

I therefore moved away from this structure to a flat directory in which the system guarantees that file names are distinct. Using this organisation (or lack thereof), it is as easy as it can be to find files, to make and edit files, and automatically to build references to files such as might be needed for HTML links. If the more structured directory is found to be useful for some purposes, it can be constructed using Unix links; at the moment, it seems to me that such access should be limited, as the files are primarily for the information system, and one would not expect them to be used directly by people or systems outside that system. Nevertheless, until such time as the information system gets big enough to look after itself adequately, it is not unreasonable that some access to the information therein by foreign agents might be necessary.

THE PROCESS FOR THE COURSES FILE.

Development was carried out by experimenting with a single course record from the Courses file. This turned out to be very successful, and the sequence of instructions produced (recorded as a macro) worked immediately without change on the complete file, and required only one alteration for what passed for perfection, so I'll use the single course record as the basis of these descriptions. I used the same sample from the Courses file as I used for my discussion of how to convert it into the intermediate file for producing the printed handbooks[4], so direct comparison is possible (though unlikely to be enlightening). Here is the sample :

```
!!number||
415.340

!!title||
OPERATING SYSTEMS (2 points)
```

```
!!prerequisites||
(415.210 and 415.231 and 415.232) or (415.212 and 415.233)

!!restrictions||
415.341

!!assessment||

!!where||
City and Tamaki

!!when||
Second semester, 3 lectures per week

!!lecturers||
Dr Alan Creak (supervisor) (50%)
Robert Sheehan (50%)

!!tutors||

!!texts required||
A.Silberschatz and P.B. Galvin, Operating Systems Concepts
(4th Edition) (Addison-Wesley) (text)

!!texts recommended||

!!description||
A computer’s operating system is the collection of software
which deals with the essential organisational tasks which
must be carried out if the machine is to deliver services to
people who need them. Its responsibilities range from dealing
with the people who use the system, to moment-by-moment
allocation of resources, to managing different activities in
progress. The paper describes the functions which the
operating system must perform, and critically examines ways
in which these requirements can be satisfied.

!!contents||
Why we need operating systems, and how they have developed.
Making the system usable: the computer-human interface.
Making the system safe: protection, security. Managing data:
memory management, file management. Managing computing:
processes, processor management, concurrent processes. Making
the system work: driving devices, scheduling.
```

As became clear in the previous analysis[4], the fields in the record have various attributes which define what sort of operations are required to produce the required output format. The attributes are :

Heading fields : The number and title combine to form the heading line for the course description, and are formatted differently from the rest of the record.

Necessity : Not all field must be present in the record; those not required should be omitted if absent or empty, but such absences of required fields (as with the !!assessment|| field in this case) should be signalled in some way. I didn't follow up this fairly important requirement in this exercise, because there was little point in putting a lot of work into the Word macro which – all being well – will never be used again, but something should be done in the eventual software. I incline towards keeping an error log, but not stopping the conversion except in cases of extreme confusion. Notice that the field can't be marked as necessary in the file; it might not be necessary for all purposes, so the necessity lies in the processing software, not in the nature of the material itself.

Tabulated : The fields from !!prerequisites|| to !!texts recommended|| are laid out in a standard tabular form following the heading.

Text : The !!description|| and !!contents|| fields contain descriptive material rather than notes for tabular presentation, so have a different format.

Segmented : The contents of a field might in some cases be presented as several units, which are in some sense to be kept separate in the final document. Paragraphs in the descriptive text, or different books in the !!texts ?|| fields, are examples. In other cases, the material is regarded as a single text string, and accidentally introduced line breaks and other sorts of white space are to be ignored.

Different combinations of the attributes appear in different fields, and, as there are different considerations in each case, that limits the degree to which uniform treatment is possible. Again, because of the temporary nature of the method, I haven't tried too hard, but it does complicate the processing.

Facilities available in HTML to reproduce the handbook formatting are limited indeed[5]. Only the rather clumsy table construct gives any significant degree of control over the relative positions of areas on the screen. Apart from that, only the various sorts of list are useful for paragraph formatting, though character styles (bold, italic, etc.) are useful for emphasis.

The Word operations which I found useful were almost all variations of the "replace all" operator. That was not a consequence of careful design, but of convenience : it is perhaps the easiest way to apply operations to every instance of some pattern throughout a complete document. This has led to some non-obvious methods, in particular those involving the "Special" style, but it works. It is possible that more efficient techniques could be worked out using more of the Word Basic operations, but I didn't think it worth spending time exploring this avenue.

(That's bad practice. One shouldn't skip things like that, but one can't do everything in the time available, and by any account this wasn't one of the most important things.)

A list of the operations which I used to convert the sample record into useful HTML follows. I don't guarantee that it will do the job perfectly in all cases, but it's fairly close. The interesting column is labelled "Comment"; that contains, among other things, the list of the real operations performed. Entries in italic type are administrative operation which don't contribute directly towards the conversion; they are likely to be specific to the Word implementation The rest of the table identifies the Word operations which I used to carry out the operations. The "mode" columns record special requirements for the replacements; "PM" means "pattern matching", and specific fonts are identified.

				Comment
Make a style called "Special".				*Used to label selected sections for later reference.*
Insert "!!end||" at the end of the file.				*For later replacement.*
Replace all –	*mode*	*by –*	*mode*	
||		||2		*Purely for convenience; "!" is a special character in pattern matching.*
!!		||1		
^p^p||1		^p||1		*Eliminate some blank lines.*
||1tutors||2^p||		||		Eliminate **unnecessary** empty fields.
||1texts required||2^p||		||		
||1texts recommended||2^p||		||		

^p||1title||2^p		*[space]*		Combine the **heading** fields to make a single header line.
number||2(*)||	PM	headera||2\1||1headerb||2\1||		Duplicate it : we need a title and a heading.
||1headera||2^p		||1headera||2^p<html>^p<head>^p<title>		Make the title and the beginning of the HTML file. The "headera" is preserved for later use.
||1headerb||2^p		</title>^p</head>^p^p<!--#include virtual="/header.html" -->^p^p^p<p>^p<b><font size=4>		A marker to fit in with the department's WWW stuff, and the page heading.
||1prerequisites||2^p		</font></b><p>^p<table>^p<tr valign = top><td width=91><i>Prerequisites:</i><td width=335>		Define the **tabulated** format, and construct the text for the entries.
||1restrictions||2^p		^p^p<tr valign = top><td><i>Restrictions:</i><td>		
||1assessment||2^p		^p^p<tr valign = top><td><i>Assessment:</i><td>		
||1where||2^p		^p^p<tr valign = top><td><i>Where:</i><td>		
||1when||2^p		^p^p<tr valign = top><td><i>When:</i><td>		
||1lecturers||2*||	PM		Special style	For a **segmented** field, treat line breaks specially.
^p	Special style	 ^p		
||1lecturers||2*||	PM		Normal style	
||2 		||2		*Surplus to requirements.*
||1lecturers||2^p		^p<tr valign = top><td><i>Lecturers:</i><td>		**Tabulated** format still.
 ^p||		^p||		*Surplus to requirements.*
||1tutors||2*||	PM		Special style	Another **tabulated segmented** field – the above sequence repeated.
^p	Special style	 ^p		
||1tutors||2*||	PM		Normal style	
||2 		||2		

||1tutors||2^p		^p<tr valign = top><td><i>Tutors:</i><td>		
 ^p||		^p||		
||1tutors||2				
||1texts required||2*||	PM		Special style	**Tabulated segmented** again : slightly different treatment. (Double line end marks the break.)
^p^p	Special style	 ^p		
||1texts required||2*||	PM		Normal style	
||2 		||2		
||1texts required||2^p		^p<tr valign = top><td><i>Texts required:</i><td>		
||1texts recommended||2*||	PM		Special style	- and again.
^p^p	Special style	 ^p		
||1texts recommended||2*||	PM		Normal style	
||2 		||2		
||1texts recommended||2^p		^p<tr valign = top><td><i>Texts recommended:</i><td>		
||1description||2*||	PM		Special style	**Segmented text**, not tabulated this time.
^p^p	Special style	 ^p		
||1description||2*||	PM		Normal style	
||1description||2		</table>^p<p>^p<i>Description:</i><p>		
||1contents||2*||	PM		Special style	**Segmented text** again.
^p^p	Special style	 ^p		
||1contents||2*||	PM		Normal style	
||1contents||2		^p<p>^p<i>Contents:</i><p>		
||1headera||2		^p<!--#include virtual="/footer.html" –->^p</html>		Add the department's footer and make the end of the HTML file.
||1end||2		^p<!--#include virtual="/footer.html" –->^p</html>		
Delete the "Special" style.				

I'm not particularly enthusiastic about using "width" to set the columns for the **tabulated** format, but I'm following last year's precedent in doing so. As it will presumably be fairly unlikely that people will be comparing different course descriptions directly, it doesn't seem to make much difference whether or not the columns are exactly the same widths. People who print the handbook might notice, but it does remove some flexibility, and isn't really necessary.

And here's the resulting HTML. Notice that it's a complete HTML file; one such is required for each course. It would probably be possible to automate the splitting and filing, but on the whole I thought it less than worth while in this experimental version.

```
<html>
<head>
<title>
415.340 OPERATING SYSTEMS (2 points)
</title>
</head>
<!--#include virtual="/header.html" -->
<font size = 4><b>
415.340 OPERATING SYSTEMS (2 points)
</b></font>
<p><table>
<tr valign = top><td width=91><i>Prerequisites :</i><td
width=335>(415.210 and 415.231 and 415.232) or (415.212 and
415.233)
<tr valign = top><td><i>Restrictions :</i><td>415.341
<tr valign = top><td><i>Assessment :</i><td><tr valign =
top><td><i>Where :</i><td>City and Tamaki
<tr valign = top><td><i>When :</i><td>Second semester, 3
lectures per week

<tr valign = top><td><i>Lecturers :</i><td>Dr Alan Creak
(supervisor) (50%)<br>
Robert Sheehan (50%)

<tr valign = top><td><i>Texts
required :</i><td>A.Silberschatz and P.B. Galvin, Operating
Systems Concepts
(4th Edition) (Addison-Wesley) (text)
</table>
<p>
<i>Description :</i><p>
A computer’s operating system is the collection of software
which deals with the essential organisational tasks which
must be carried out if the machine is to deliver services to
people who need them. Its responsibilities range from dealing
with the people who use the system, to moment-by-moment
allocation of resources, to managing different activities in
progress. The paper describes the functions which the
operating system must perform, and critically examines ways
in which these requirements can be satisfied.
</table>
<p>
<i>Contents :</i><p>
Why we need operating systems, and how they have developed.
Making the system usable: the computer-human interface.
Making the system safe: protection, security. Managing data:
memory management, file management. Managing computing:
processes, processor management, concurrent processes. Making
the system work: driving devices, scheduling.

<!--#include virtual="/footer.html" -->
</html>
```

There is one error in the file : the Assessment and Where lines are concatenated. That's because the omitted Assessment field wasn't detected; with a record carrying a valid Assessment field, all was well.

THE PROCESS FOR THE ORDINARY HANDBOOK MATERIAL.

In comparison with the acrobatics required to convert the Courses file into HTML, the ordinary handbook text was almost trivially easy. (For once, "facile" might be the right word, except for its connotation of deceitfulness.) I think that there are two reasons for this :

- In defining the ordinary material, I've carefully avoided any challenges. I have given my excuses for this above, but I think it's honest enough. Certainly the ordinary part encompasses most of the handbook material, apart from the course descriptions.

- More specifically, it is quite easy to identify the parts which must be presented in certain ways, and to identify them with suitable mark-up symbols. It helps that the passages requiring special formatting are mostly lists, which fit in well with the HTML lists, or tied to the document format in that they are chapter or section headings.

In practice, that final point turns out to be not quite as simple as it might sound. The result of making that assumption in the experimental run was to produce far more sections than I really wanted; some later chapters of the handbook are composed of a lot of little fiddly points (hints and tips about general facilities available, where to find things, and such) most of which are about one or two lines long and really don't warrant the status of sections. I identified them as sections because they have numbers in the original versions of the handbook, but that was probably a mistake; they weren't listed as sections in the old contents, and the first lines weren't formatted as section headers. Perhaps the numbers weren't necessary (I don't much like them anyway), but someone must have wanted them at some time; the example suggests that numbering shouldn't be inextricably tied to document structure, though obviously association between numbering and structure can be important and should be easy to set up.

(In a later change, I recast this material by collecting the separate items into a single HTML file for the chapter as a whole. I kept the separate "section" names in the index, because they showed what sort of information was available, but linked them to appropriate positions within the chapter file. This works much better. If I want to preserve this sort of structure, yet more mark-up language is required; is it justified ?)

There are a few instances in the ordinary parts of the handbooks of text inserted from other files. The course descriptions are the most notable examples, but the graduate handbook also contains lists of academics' publications and research interests, and other cases are likely as the information becomes more coordinated. I managed all these manually in this experiment, but for regular use some sort of "subroutine call" is required. Notice that it is not sufficient simply to make a "final text file" at the mark-up level including all the handbook text, because the content of printed text and HTML files is different. In illustration, consider the lists of stage I, II, and III lecture courses in the undergraduate handbook : in the printed versions, there is in each case a passage of descriptive text, but this is followed in the printed text by full course descriptions and in the HTML version by a list of links to the separate course descriptions. All in all, I think it's fair to say that this exercise has shown that the ordinary parts of the handbook can be handled quite well, though it might be that I need some more syntax to define a few additional facilities.

THE REST.

I have worked through almost all the rest of the material by hand, and I am reasonably confident that there are no insuperable difficulties. Perhaps the interesting problem is to invent mark-up notations which convey the required information in effective form without being unreasonably obtrusive. Tables inevitably loom large; very simple conventional tables shouldn't be hard to convert into HTML, though converting into acceptable Word formats might be harder. (Setting the column widths is the awkward bit.) Tricky formatting with nested tables might be a lot harder, and require a lot of markup; if the object is simply systematically indented lists and sublists, definition lists might be an alternative worth investigating. Notice, by the way, that the table used to present the formatted lists at the heads of the course descriptions were comparatively easy to manage, as they're not required to look like tables; tables which are really tables might well be harder.

A special case is of some interest. The Contents list was constructed automatically by selecting the chapter and section headings from the text file and applying appropriate massage, following much the same lines as those used in other cases this year. They depended wholly on Word operations, but are even less worth preserving than some others because this operation should not be needed at all; the contents should come directly from the Skeleton file[2], but there wasn't one this year, except in the mind. Of more interest is the observation that the file format chosen earlier made it possible to insert the HTML links to the files for the different sections completely automatically, and it worked without a hitch.

CONCLUSIONS.

There is no big conclusion – except, perhaps, that the conversion into HTML is quite manageable provided that the requirements are clear, and that in turn can be arranged by providing appropriate information in mark-up form. Three particular points stand out as requiring attention soon :

- It is important to sort out the basic data into a well designed set of files. The data as I first received them were stored in many separate files rather arbitrarily, but in all cases the divisions were based on the layout of the handbook rather than the nature of the data, and that led to considerable difficulties in case one wished to change anything uniformly. My reorganisation into files for Courses, People, and so on[2] is certainly a useful step forward; here, I've suggested that the handbook material can also usefully be seen as a few parts with rather different natures. This doesn't necessarily mean that physically separate files are required, but it does suggest that there will be different forms of treatment somewhere in the system. (The different files were convenient for the manual conversion used this year with Word, but if I move to the sort of production line form I suggested[4] the same arguments don't necessarily hold.)

- A careful assessment of just what HTML features are to be used is required. There are several cases where I wasn't sure what to do; I've mentioned the little-bitty-sections question, and the how-to-specify-HTML-links question. A good solution to the first so far as the presentation is concerned turned out to be to replace individual section files by links into a single HTML file for the whole chapter; automating that wouldn't be difficult, but does require some new mark-up symbols, and a rather cleverer processor which knows whether or not to make HTML files at section or chapter level. My best "solution" to the links problem is just to accept the links in ordinary text form in the mark-up files, and hope that they're properly made, but this raises some difficulties in preserving the links through the conversion back from the handbook form where the links are removed.

- It will be necessary to design some mark-up syntax for including material from other sources. Examples noted here have been the Courses and People files, and more generally access to different sorts of data held in different places will be useful. In keeping with the nature of mark-up languages, a notation with a declarative appearance is appropriate, though in practice it is certain to be interpreted actively. For example, the handbook requirement that a list of certain course descriptions should be inserted at a particular place will be interpreted as an instruction to insert the complete set of descriptions when making the printed handbooks, but as an instruction to insert a list of links in the HTML version. A possible form might be

  !!represents stage2courselist||

  As well as text, notation for including graphic material is required. Again, this will be used in different ways in different cases – for the printed handbook, the final form is an instruction to insert the required picture nearby, while for the HTML form a link to a .gif file is required. These requirements are much less demanding than those for text inclusion, which suggests that a different syntax might be appropriate. A possibility is

  !!picture Tamaki||

  which can readily be interpreted in either way given sensible names for the files.

REFERENCES.

1 : E. Hillary, asked why he wanted to climb Mt Everest (folklore).

2 : G.A. Creak : *Background for a document generator*, unpublished Working Note AC115 (September, 1997).

3 : G.A. Creak : *Going back again*, unpublished Working Note AC119 (November, 1997).

4 : G.A. Creak : *Designing the document factory*, unpublished Working Note AC117 (November, 1997).

5 : D. Raggett : *HTML 3.2 Reference Specification* (World Wide Web Consortium, REC-html32, January, 1997).