`curl`: Automated Linking of Pages in a Web Document

Andrew Davison

Technical Report 96/23
Department of Computer Science
University of Melbourne
Parkville, 3052, Australia
Email: ad@cs.mu.oz.au
URL: http://www.cs.mu.oz.au/~ad

May 1996

Abstract

Large Web documents cause problems for their users, and for their authors. The most common user problem is getting lost, due to the convoluted organisation of the document, and poor navigational and orientation support tools. The author's main problem is deciding on a suitable organisation, which is expressive enough while not creating an impossible burden of building and maintaining links.

These problems are addressed by `curl`, a tool for automatically creating the links between the pages of a Web document. `curl` utilises the metaphor of a book to organise pages, which is both powerful and familiar. Typically, every page is linked to its nearest siblings, its parent, and the top page of the document. Pages can contain two kinds of contents lists, and there are special pages for lists of links to keywords, figures, tables, and new pages. There is also a search engine available from every page.

`curl` is still being developed (version 2 is the current release), and so this article also contains a discussion of features which will be added to `curl` in the near future. More speculative extensions are also briefly described.

1 Introduction

As the size of a Web document increases, the recommended approach is to subdivide the document into pages, with each page in its own file. The task is then to link the pages back together in a way that is usable by a reader, and manageable by the author. Such documents can consist of 10's or 100's of Web pages, and are often extremely difficult to use and maintain. This is *not* due to the HTML coding of the information on the pages, but rather because of the complexity of the links *between* the pages.

Large Web documents cause distinct kinds of problems for users and authors, and we shall briefly consider each group. This discussion will motivate section 2, which is an overview of the features of `curl`, a tool for automatically generating the links between the pages of a Web document. Section 3 looks at features that will be added to `curl` in the near future, while section 4 examines more speculative extensions. Section 5 closes with a summary of the problems associated with large Web documents, and the ways that `curl` helps to address them.

1.1 User Problems

The most common user complaint is of being 'lost', which can be divided into three specific problems: not knowing where to go next, knowing where to go, but not knowing how to get there, and not knowing where they are in the overall document structure [Edwards and Hardman 1989; Van Dyke Parunak 1991].

There are two main reasons for these problems. The first is a lack (or loss) of coherence; coherence is the user's ability to form a mental model of the information that he or she is processing [Thüring at al. 1995]. A breakdown of coherence can occur at a local level, when the user is unclear how a page is related to the ones around it, or at a global level when the structure of the document is ill-defined. This is also known as the 'Art Museum Problem' [McAleese 1989], which is the inability of the user to build any more abstractions about the document after browsing for a long period.

Another reason for user confusion is cognitive overload, which occurs when there is too much to remember about the browsing task [Conklin 1987]. A common instance of this is the 'Embedded Digression Problem' [McAleese 1989], when a user follows so many alternative chains of thought (encouraged by the links in the document) that the original goal is lost.

Problems with coherence and cognitive overload can be lessened by improving the orientation and navigational aids in the Web document. For instance, a history trail could be used to record previous pages visited (in chronological order) and their context within the document. Existing browsers, such as Mosaic and Netscape, have a history list mechanism, and buttons for moving forwards and backwards through it. However, Jones and Cockburn [1996] have found these aids to be misleading to users, and Catledge and Pitkow's [1995] empirical studies indicate that few people use the history list.

A more crucial aid for users is the presence of a powerful metaphor for representing the document (i.e. the collection of Web pages) and the pattern of links between the pages. One way of evaluating such a metaphor is to apply the criteria suggested by Garzotto et al. [1995] for hypermedia applications. These seven criteria are: richness, ease of use, consistency, self-evidence, predictability, readability, and reusability.

Richness is the amount of information that can be represented, and the ways it can be accessed. Ease of use measures the accessibility of information, and the understandability of the operations. Consistency can be summarised by the rule 'Treat conceptually similar elements in a similar fashion, and conceptually different elements differently.' Self-evidence expresses how well users guess the meaning of the content and navigational aids. Predictability measures how well users anticipate an operation's outcome. Readability is related to how easy it is to understand the document, which depends on its content and organisation. Reuse considers how easily it is to reuse content and navigational tools in different contexts, and for different purposes.

A metaphor based on the layout and organisation of books (the 'book' metaphor) satisfies most of these criteria.

Many of the navigational features in books are based on the hierarchical decomposition of the contents, into chapters, sections, and so on, which encourages levels of abstraction. Books also contain contents lists, lists of tables and figures, and indexes, which facilitate a variety of search techniques. Page numbers also encourage various search strategies, from linear search to divide and conquer, often helped by the logical numbering of chapters, sections, and other components.

Popular enhancements to the traditional book format include chapter summaries, side bars, and the use of coloured text and graphics. Often the first chapter of a book outlines several different reading orders for later chapters, based on the interests of the audience (e.g. undergraduate, postgraduate, executive), and sections are marked to indicate that they can be skipped upon a first reading.

Books can be personalised to a certain extent, by annotating the text (e.g. writing comments in the margin), or by the use of bookmarks.

The 'book' metaphor has proven its worth over hundreds of years, and is consequently extremely familiar. It is also very flexible, encompassing a wide range of different contents and organisational styles, such as lecture notes, newspapers, encyclopaedias, catalogues, and dictionaries. Even supposedly unstructures formats, such as the essay, obey many structural rules.

Of course, there are problems with books in the printed form. Perhaps the most serious is that they are difficult and slow to search (when compared to text in the electronic media). For instance, the traditional index is useful, but will not contain every word, and cannot deal with synonyms. Also, it does not scale well, unless the book is highly organised, as in an encyclopaedia. Studies have shown that users prefer to use electronic search when they have a choice between it and a manual index, and even find search based on simple string matching useful [Rada 1991]. Many users look for material by first searching then browsing; these phases are sometimes called "What to Where" and "Where to What".

Books are hard to update, although manuals in ring binders have tried to address that weakness. A related problem is the difficulty in arranging the parts of a book for the particular needs of a reader. For example, the introductory material in a foreign language text is less necessary when the book is being used as a reference.

It is difficult to reuse parts of a book in different contexts (e.g. utilise an example both in the introduction and in a later chapter) without simply duplicating the material.

It is not easy to record a trail of pages, sections, and so on, which have been studied. Most readers resort to annotating the table of contents. Bookmarks, although useful for marking a few pages, do not scale well.

Ordinary books lack interactivity in the sense of animation and sound. On a more serious level, it is not possible for a printed text to accept reader's questions and respond. This drawback is related to the difficulty of 'tuning' a book to a reader's needs, since responses could only be generated dynamically, during the reader's interaction with the book. Activity books are an interesting response to the lack of interaction in traditional books.

In summary, the 'book' metaphor addresses the twin problems of loss of coherence and cognitive overload. While this metaphor offers a rich set of organisational, orientation and navigational constructs, it still has some limitation. Fortunately, many of these can be overcome by applying the metaphor to text in the electronic media.

1.2 Author Problems

The first problem confronting the author of a large document is how to subdivide the material into manageable parts. In the context of Web documents, each of these parts usually becomes a separate Web page in its own file. A related issue is how to link those pages back together to form a coherent whole.

The 'book' metaphor is a good solution to both of these problems since it offers a range of useful structuring rules. This claim will be illustrated through reference to the Web document in Figure 1.

The document is hierarchical, hence its tree-like form, with each node corresponding to a page of the document. `newsletter` introduces the newsletter, and contains links to the `editorial` and `humour` sections, and to the `related-info` page. The `editorial` and `humour` sections consist of a series of related pages which are reached by links.

Building this Web document requires an enormous amount of work, because of the large number of page links needed to support browsing over related topics, and movement between sections. For instance, `pathology` should have links to `marks`, `whome` (related pages), `editorial` (the section header), and `newsletter` (the overall contents page). `editorial` should have links to `marks`, `pathology`, `whome` (its articles), `humour` and `related-info` (other sections), and `newsletter` (the contents page).

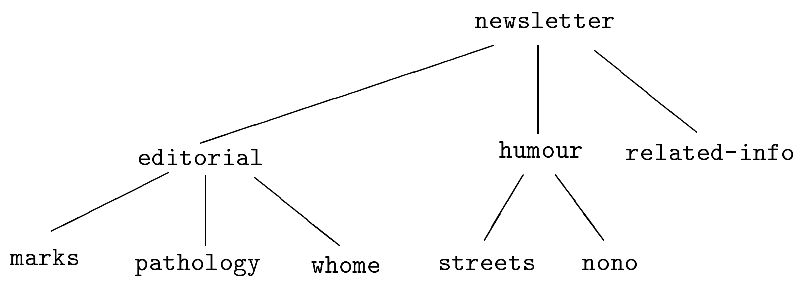


Figure 1: A Web Document

Building these links is error-prone; changing them is probably even harder to get right. For example, if `pathology` was moved after `whome`, then some or all of the links in `marks`, `pathology`, `whome` and `editorial` would need to be changed.

Documents of this kind often have separate pages listing the tables, figures, keywords, and new pages. These features increase the link complexity even further.

As the size of a document increases, it becomes harder to find things even when the document has a keyword index and a hierarchical structure. Therefore a search engine is essential for scanning the document and dynamically generating a page of links to relevant pages. This links page can be viewed as a simple kind of 'personalised tour', based around the user's interests.

The clarity of a document is often improved if the organisational parts are distinct from its content. This separation can be encouraged by realising that the organisation a in Web document is represented by page links which fall into two categories.

Links derived from the 'book' metaphor are mostly independent of the content. For instance, every page will have links to the previous page, next page, parent page, the keywords index page, and so on. The meaning of 'previous page' (and other navigational features) can be understood in terms of the page's position in the document without understanding (in detail) its contents.

However, there will always be links which are content specific. For example, the phrase 'Street Lighting' may link several pages which discuss this topic. Unfortunately, because of the domain-specific nature of these links, they place a higher load on the coherence and cognition of the reader when using them. Therefore, they should not play a major role in the document structure [Alschuler 1989].

The discussion of the 'book' metaphor in the previous subsection highlighted its support for a wide range of orientation and navigational aids. However, the author should be able to choose which of these mechanisms to utilise, and which to leave out. For instance, a novel almost never requires an index, but sometimes has a list of chapter headings.

As a document grows, it becomes increasingly difficult for the author to remember all of its details. Indeed, many documents are written by groups (e.g. newspapers, conference proceedings) and the editor, although having overall control of the structure and content, may not be familiar with all the articles. Therefore, it is necessary for him to have tools for analysing the document.

2 `curl`'s Features

`curl` is a tool for automatically generating the links between the pages of a Web document. `curl` utilises the 'book' metaphor through a contents file containing the file names of the individual pages. `curl` modifies each file to include the necessary links to its neighbours, parent, initial page, and to pages listing the figures, tables, keywords, and new pages in the document. These figures, tables, keywords, and new additions pages are generated automatically. `curl` can also produce two kinds of contents lists, which can be inserted into pages.

The contents file for the document in Figure 1 is:

```
newsletter {
  editorial {
    marks
    pathology
    whome
  }
  humour {
    streets
    nono
  }
  related-info
}
```

Each name corresponds to the HTML file which holds the document page (e.g. the `marks` page is stored in `marks.html`). Although the pages are assumed to be correctly written in HTML, they do not need to include any links which reflect the document format defined by the contents file (there may be links, but these will be for other purposes).

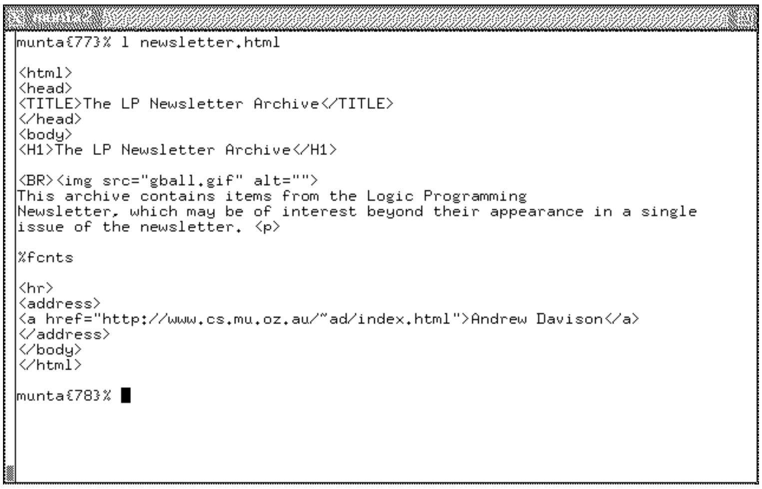
An interesting way of viewing the contents file is as a form of document schema, or template.

This example can be found at:

`http://www.cs.mu.oz.au/~ad/curl/example/`

It is derived from a medium sized document (consisting of over 350 pages) which starts at:

`http://www.cs.mu.oz.au/~ad/alp/archive.html`



```
munta{77}% l newsletter.html

<html>
<head>
<TITLE>The LP Newsletter Archive</TITLE>
</head>
<body>
<H1>The LP Newsletter Archive</H1>

<BR><img src="gball.gif" alt="">
This archive contains items from the Logic Programming
Newsletter, which may be of interest beyond their appearance in a single
issue of the newsletter. <p>

%fcnts

<hr>
<address>
<a href="http://www.cs.mu.oz.au/~ad/index.html">Andrew Davison</a>
</address>
</body>
</html>

munta{78}%
```

Figure 2: The original `newsletter.html`

The directory holding the various HTML files (`newsletter.html`, `editorial.html`, etc) is called `example`. The directory must contain a contents file, which is called `lpn-contents` in this example (but any name can be used).

The original `newsletter.html` file is shown in Figure 2. After it has been modified, `netscape` displays it as in Figure 3. The only non-HTML feature in the original file is the `%fcnts` command, which causes `curl` to insert a full contents list.

Note that the previous and next icons (the left and right arrows) have red crosses through them (they may look black in this article, but they really are red :)). This means that there is no previous or next page links for this page. Also, if the `Up` or the `Home` link icons are clicked upon, they return to `newsletter.html` since it is the top level page of the document.

Clicking on `Editorials` in the contents listing, causes `netscape` to display the contents of the modified version of `editorial.html`, as shown in Figure 4. Its `Next` link icon (the right pointing arrow) goes to `humour.html`, which is the next subsection in the document.

The `Up` and `Home` links can be viewed as ways for the user to seek an overview of their current location by moving up the hierarchy. The parent pages supply an overview by using the contents operations: `%fcnts` and `%cnts`. `curl` processes `%cnts` by generating a (textual) map of the hierarchy one level below the current node. `curl` processes `%fcnts` by building a (textual) map of all the levels below the current one. Clicking on an entry in one of these contents lists gives the user a way of focusing on a given element.

The `Prev` and `Next` links guarantee that movement from one sibling page to another is always reversible, which is an important piece of predicatable behaviour. Another way of understanding these actions is as lateral movement

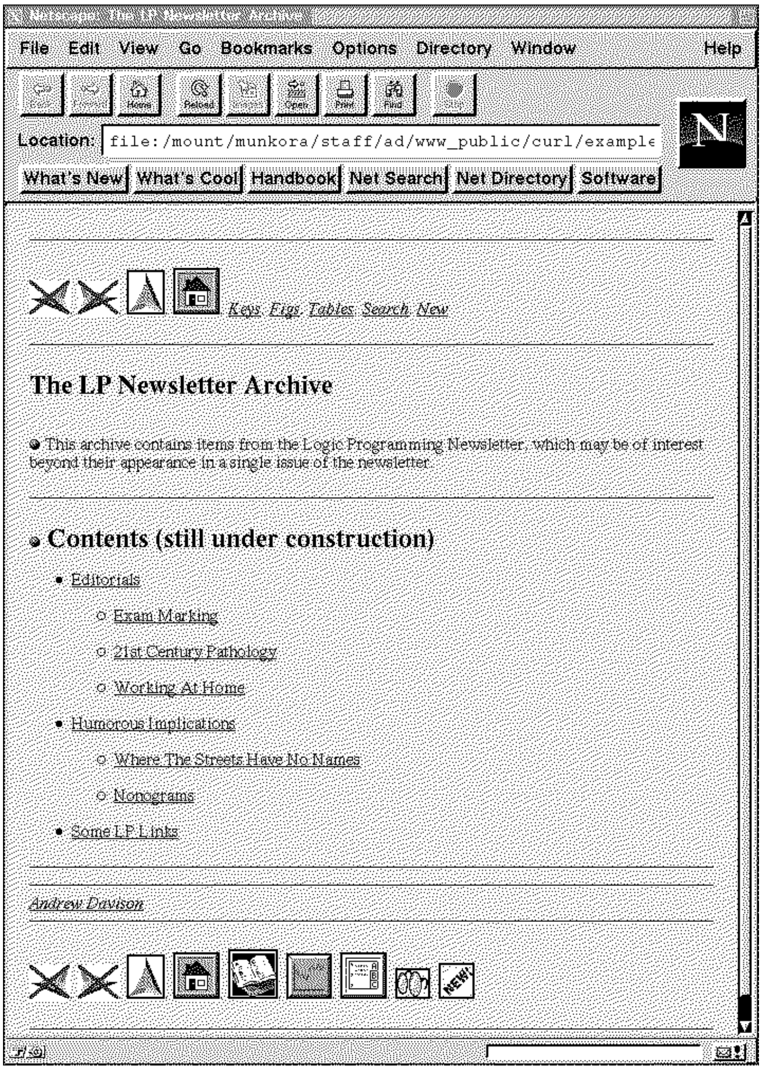


Figure 3: The new version of `newsletter.html`

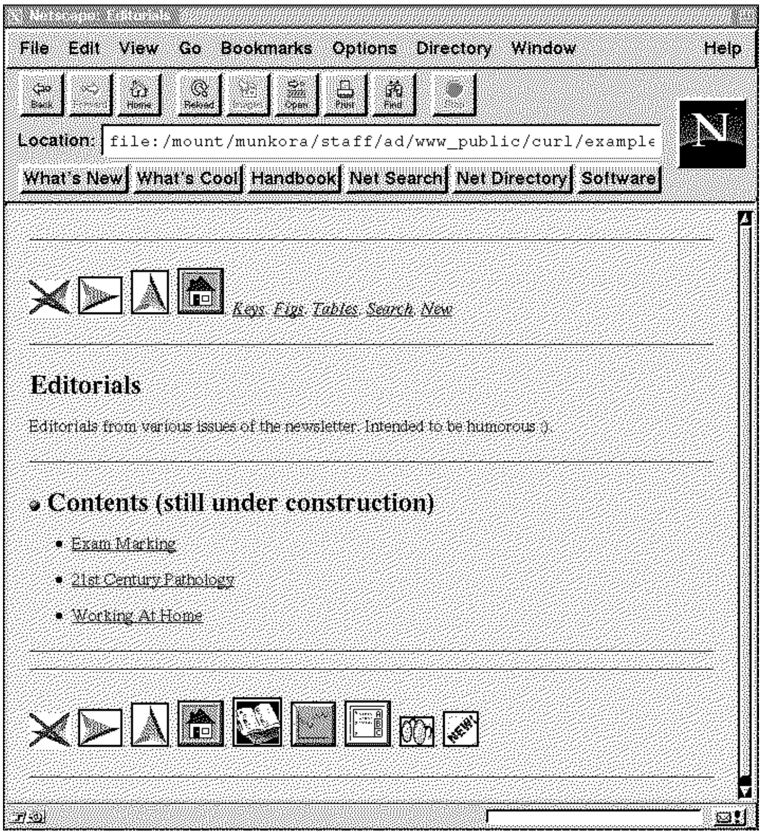


Figure 4: The modified `editorial.html`

operations across one level of the hierarchy.

Important omissions from these navigational tools are support for history trails, trees and bookmarks. Currently, a user must utilise the browser if those features are requires. This is far from ideal, and improvements to this situation are discussed in section 4.

2.1 Support for Incremental Link Change

Changing links is simply a matter of reordering the contents file and reexecuting `curl`. The out of date links are removed and new ones generated. Adding or removing extra pages from the document is achieved by adding or removing their file names from the contents file.

This addresses the common requirement to keep updating Web documents, which occurs much more frequently than with printed texts. An important consideration is the cost of such updates, which may involve the addition of one or two new pages to a document of several hundred pages. `curl` uses changes to the contents file and the modification dates of files to decide which files to reprocess. This is not always ideal, but a lot less costly than examining the contents of each file.

2.2 Hierarchical Structuring

The hierarchical nature of the pages in the document can be used to organise the HTML files. For instance, `marks.html`, `pathology.html`, and `whome.html` can be placed in a subdirectory called `editorials`, and `streets.html` and `nono.html` can be moved to `humour`. This file arrangement is specified in the contents file with the `path` command:

```
editorial {
path = editorials
  marks
  whome
  pathology
}

humour {
path = humour
  streets
  nono
}
```

Note that `editorial.html` and `humour.html` are still in the `example` directory. A `path` command's scope starts from the beginning of the '{' preceding it, and extends to the matching '}'.

```
munta{112}% l streets.html

<html>
<head>
<TITLE>Where The Streets Have No Names</TITLE>
</head>

<body>
<H1> Where The Streets Have No Names</H1>

<b>Keywords</b>: street, map. <p>

There exists a city where all streets are exactly north-south or
east-west. Because of that there are often many different routes
from A to B with the same length. <p>

The map below represents a part of this city.
How many shortest routes are there from A to B? <p>

<img src="street.gif" alt=""> <p>
<b>Figure</b>: A Street Map. <p>

</body>
</html>

munta{113}%
```

Figure 5: The original streets.html

2.3 Recording of Keywords, Figures and Tables

The 'book' metaphor includes a diverse range of search techniques, including the index, and lists of figures and tables.

The files being processed by curl can contain keyword information, and figure and table captions. These are collected into three special HTML files and curl builds links between them and their original files.

Links to these special files can be included at the top and bottom of every page, as in newsletter.html and editorial.html in Figures 3 and 4.

The original version of streets.html is shown in Figure 5. Note the use of an image which is labelled with the curl figure command Figure:. When curl processes this file, the string following the Figure:, up to the '.' (i.e. A Street Map) will be added to the figs.html file. As the name suggests, figs.html holds links to all the figures in the document.

In a similar fashion, if streets.html contained a table it could be labelled with the Table: command which would cause curl to add its string to the table.html file. table.html holds links to all the tables in the document.

streets.html also illustrates the use of keywords in curl. Near the top of the file is the line:

```
<b>Keywords</b>: street, map. <p>
```

This will cause curl to add street and map to keys.html as keywords.

Default keys.html, figs.html, and table.html files are included in the curl release, but an author can customise them.

2.4 Link Style Files

The format of the automatically generated links is defined in style files, allowing the format to be modified. By default, links are added both to the top and bottom of a file, and so there are separate style files for these two cases.

Link styles can be applied to a subset of the pages of a document, allowing different subsections of a document to be linked differently.

Links can be represented either by text (e.g. `Next`, `Up`) or by user-specified icons.

It is not generally useful to remove the `%prev`, `%next`, `%up`, `%home` and `%search` labels from a style file since it restricts the navigational capabilities of the resulting Web document. However, if the document does not contain any keywords, figures, tables or new pages then `%keys`, `%figs`, `%tables` and `%new` can be usefully deleted.

Link style files encourage a consistent approach to navigation across pages, and the links are inserted in such a way that they are clearly separate from the page's content.

Style files can be modified, but were designed so that all sibling pages below a given parent must be represented in the same way. For instance, it is not possible for `streets.html` and `nono.html` in the `newsletter` example to use a different navigational style.

2.5 A Search Engine

A search engine is essential as the size of the document grows, since it becomes increasingly hard to find things, even when the document has a keyword index and a hierarchical structure.

Every document has its own search page, called `search.html`. Given a set of search strings, this page invokes a CGI script (written in C for speed) which searches over the HTML pages of the document. It dynamically generates a page of links to the pages containing some or all of the search strings.

Activating the `Search` link on a page loads `search.html`, which is shown in Figure 6 after the user has entered the strings 'street' and 'map'. When the `Start Search` button is pressed, the search engine generates the page of links shown in Figure 7.

The copy of `search.html` in each document needs to be configured before it will access the search engine correctly.

The search engine represents the first departure from the 'book' metaphor, in order to take advantage of the electronic medium.

An interesting view of this search engine is as a way for the user to generate a simple kind of 'personalised tour' of the document, based on strings of interest. An advantage of such a tour is that the list of links go to pages within the document rather than to copies extracted from their context. A disadvantage is that the search results page is only temporary, its continual existence dependent on the caching characteristics of the browser. In any case, the page will be lost when another search is performed.

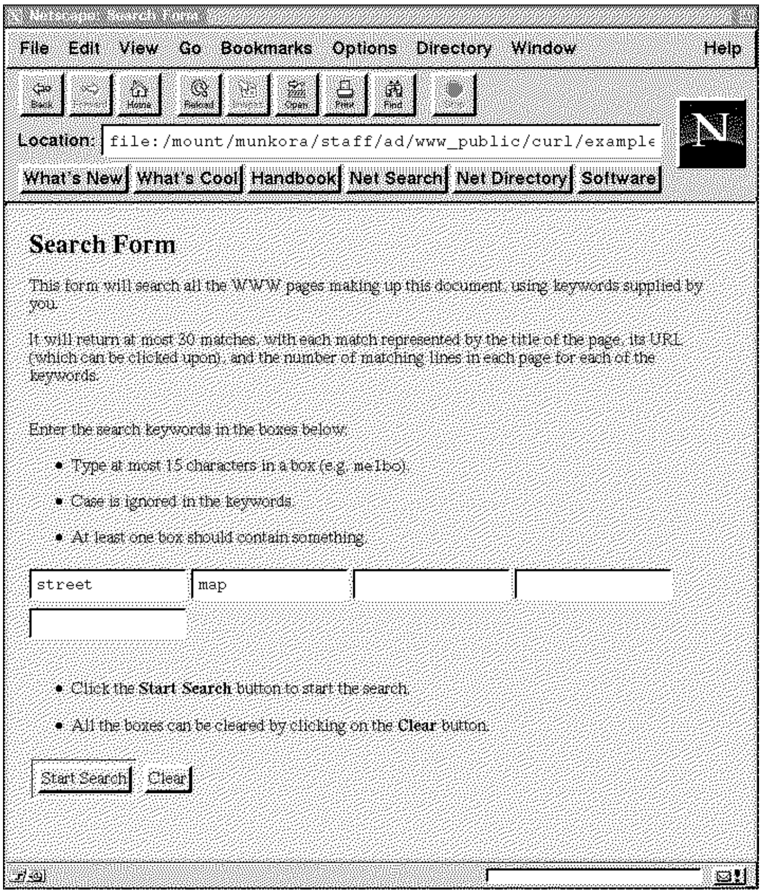


Figure 6: `search.html`

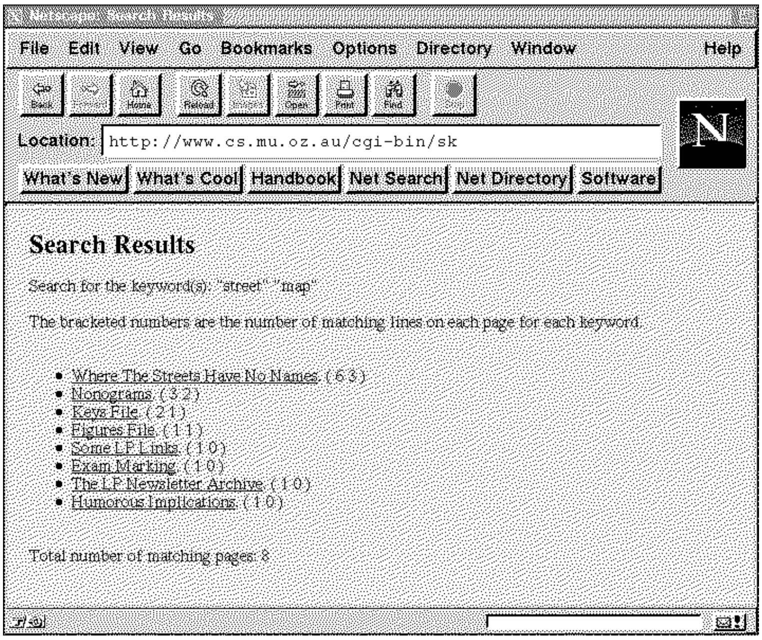


Figure 7: Search Results

A string-based search is rather simplistic, but it works well for documents of less than a few hundred pages. The search results are ordered using a weighting scheme based on how frequently the strings occur in each file.

2.6 A New Additions Page

New pages can be marked with a '*' in the contents file, which causes a link to the new page to be added to `new.html`. This 'New Pages' file can be accessed from any page via a link at the top and bottom of the page. This permits a frequent user to discover new pages without a lengthy search.

Any contents lists which refer to a new page will be marked with the `new.gif` icon or the word '`NEW!`'.

2.7 Easy Removal of `curl`-generated Links

A possible concern for a `curl` user is the way it changes HTML files without making backups. However, these changes are almost all additive – none of the text of the original HTML file is deleted apart from the `%fcnts` and `%cnts` commands which are replaced by contents listings. Also, the changes can be easily reversed by calling `decurl`, a utility included with the `curl` distribution. `decurl` can be applied to a single file, to all the files in a single directory, or to every file in the document.

2.8 Tools for Analysing Word Occurrences

The `curl` release includes utilities for collecting information about how frequently words occur in pages. This is useful for deciding on keywords, and for refining the document hierarchy. For instance, pages that make frequent references to the same words should perhaps be grouped together.

The main word occurrence tool can also be used as a search engine by the author, without requiring the CGI search script and its browser interface.

2.9 Tools for Manipulating Links Outside the Document Hierarchy

The hierarchy of the `newsletter` example in Figure 1 reflects the fact that `mark`, `pathology`, and `whome` are all editorials, while `streets` and `nono` are humorous pieces. However, there may be other ways of relating some of these articles, which it would be helpful to include in the hierarchy.

For example, if `whome`, `nono` and `related-info` mention a common topic, such as street lighting, then it may be useful to link them together, as in Figure 8.

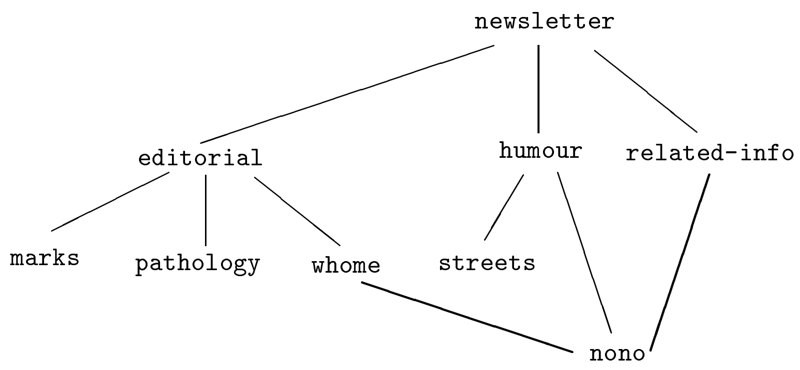


Figure 8: The Extended `newsletter` Document

This cannot be done by `curl`, but the `curl` release comes with tools for facilitating this kind of linking.

These links are quite different from the links manipulated by `curl`, since they are domain-specific.

3 Planned Features for `curl`

This section describes a number of features which may be added to `curl` in the near future, in order to augment the basic 'book' metaphor. More speculative changes to `curl` are outlined in section 4.

3.1 Printing

Many studies of hypertext and Web users have highlighted their wish to print out the documents being browsed. Our response initially involves the linearisation of the Web document structure into a single HTML file. This can be quite difficult in the general case, but the 'book' metaphor makes it simple for `curl` documents. The translation of the keywords, tables, figures and new additions pages are slightly problematic, since their links must be replaced by meaningful references based on a numbering scheme for the document's headings, figures, and tables.

Once a single HTML file has been generated, existing tools can be used to convert it to a suitable printable format, or it could be printed via a browser.

One issue is whether parts of a document can be printed. Printing a single page is trivial, but larger groupings require linearisation beforehand.

The choice of what to print will be dictated by the author. This means that the linearisation can be performed by the author while the Web document is being put on-line. Restricting the user in this way is a limitation but is appropriate for many kinds of Web information, such as lecture notes and work-in-progress.

3.2 Guided Tours

A guided tour is a collection of pages from the Web document which highlight a particular theme. For instance, a Web document about Melbourne, Australia may have a guided tour about the cultural activities, and another about the beauty spots.

Guided tours reduce the user's cognitive load since they offer a ready-made sequence through the document. They can also be restrictive if it is not possible for the user to follow their own lines of enquiry off the prescribed path.

The guided tours created using `curl` will be developed by the author as the document is being prepared. The tours will be accessible from any page in the document, in the same way as the keywords, tables, figures, new additions, and search pages. Tools will be available to generate a page of links like the results page generated by the search engine. The author may then choose to edit the page to add some extra background material, or other content.

The result will be a guided tour consisting of links to pages still within the document, which will facilitate browsing off the tour path. Also, the availability of a tour from any page in the document will allow a tour to be resumed easily.

A drawback of this approach is that it precludes the development of tours by the user, except by means of the search engine.

3.3 Extended Search

The search engine included with `curl` is too slow when there are more than a few hundred pages in the document. This could be remedied in a number of ways. For instance, the search engine could have the optional ability to restrict its analysis to the title lines of each page. A less easily implemented strategy would be to restrict the search to pages containing contents listings.

Another avenue worth investigating is to have the author generate an inverted index just prior to the Web document going on-line. Any searches could utilise this to obtain their matches [Carlson 1989].

The range and complexity of search queries could be extended, although there is little evidence that users actually need or want richer capabilities of this kind.

3.4 User Feedback

A feedback page would allow a user to enter their name, address, and comments into a form, whose input would then be sent via e-mail to the author. This mechanism would require a CGI script to be installed, and for the feedback form to be configured, in a similar way to the search engine and its page.

3.5 Richer Navigational Links

The siblings at a given level in the hierarchy are linked in a two-way list configuration. Some document may be better formatted so that the siblings are in a loop, where the last page is connected back to the first.

When a user accesses one of the index pages (`keys.html`, `figs.html`, `table.html`, `new.html`, or `search.html`), there is no explicit link back to the page where the user came from. This is not easy to do with the current `curl` mechanism, since it requires a link which is determined at run time. However, it should be possible to have links between the index pages, and a link from each to the home page of the document.

4 More Speculative Features

The features described in this section are derived from interesting research topics as they might be applied to `curl`.

4.1 Typed Links and Pages

The navigational links added by `curl` could easily be typed. Every page has a 'previous' link, a 'next' link, an 'up' link, and so on, which always perform similar roles. In addition, the default `curl` pages in each document (`figs.html`, `keys.html`, etc) always have the same meaning, and so could be typed.

Allowing a richer set of types would permit the document to be structured in a clearer manner. For instance, a document about actors would consist of instances of the actor type page. Each actor page would contain links to instances of the film bibliography page, a quotes page, and a pictures page. The links between the page instances could also be typed.

Typing would have immediate benefits for navigation, which could utilise types to present a variety of abstract views of the document in addition to the two kinds of contents list currently available. Search would be improved by using type information as an extra source of guidance.

Once pages and links have types, it is possible to utilise design methodologies, such as RMM [Isakowitz et al. 1995], to help build documents. Types also have interesting connections to semantic nets and AI-based inference techniques [Rada 1991].

4.2 Separating Links from Content

`curl` adds navigational links to Web pages, but an alternative approach is to generate a separate navigational layer. This idea was developed by Bruza in his hyperindex and hyperbase system [Bruza 1990].

An advantage of this separation is the ease of reusing content in different contexts. For instance, the pages in the document could be linked in one pattern for novices, and another for advanced users. A similar idea would be to offer a repair manual with a number of different views: one for training, one for routine maintenance, and yet another for troubleshooting.

Once links are separated from content, it becomes much easier to dynamically create or change links, as outlined in the next section.

4.3 Dynamic Processing

Dynamic processing refers to the creation of pages, documents, or links at run time, typically in response to the user's activities. In fact, the key advantage of dynamic processing is its focus on the user rather than on the author.

`curl` contains one dynamic processing element: its search engine, but the search results page is only transitory. A truly dynamic page generation system would allow such pages to be encorporated into the user's personal version of the document. The hypertext system, Intermedia, has such a mechanism for creating views of a document dynamically, based on a search which analyses keywords attached to links [Garzotto et al. 1991].

Another dynamic mechanism, familiar to browser users, is the history list, and buttons for moving forwards and backwards. These are not easy to understand since they are related to the stack of recently visited pages rather than to a temporal sequence [Jones and Cockburn 1996]. Another problem is that the list is not specific to the document. Also, a history list is not sufficient since the context of the visited pages is just as crucial for improving the user's orientation [McAleese 1989].

The availability of a detailed history list for the document would benefit on-line help, tutorials, and guided tours, which could utilise knowledge about the pages visited, and (implicitly) the ones still unseen.

4.4 Extending the 'book' Metaphor

On-line help and tutorials are two features of the 'book' metaphor not yet utilised by `curl`. However, they are so domain-specific that it may be difficult to develop a useful framework for them within `curl`. The same may also be true for glossary and bibliography pages.

Another interesting aspect of the 'book' metaphor is the reading plan, which typically takes the form of a dependency graph between chapters. This can be viewed as a guided tour based on the development of ideas in the book.

The 'book' metaphor spans a wide range of different book types. Essays, newspapers and repair manuals all employ some aspects of the metaphor, but with very different results. Perhaps a classification, or hierarchy, of book types will suggest different variations of `curl`?

5 Summary

The main problem for users of large Web documents (and hypermedia systems) is becoming lost, which is a result of loss of coherence and of cognitive overload.

A key problem for an author is the time-consuming task of linking pages together to form a document with a clear navigational structure. The subsequent task of maintenance is also very difficult.

`curl` addresses these problems by automatically linking Web pages based on the 'book' metaphor, with its rich, familiar set of navigational and orientation aids. `curl` also supports incremental change to a document, with an emphasis on efficiency.

`curl` comes with a set of default pages for storing information about keywords, figures, tables, and new additions to the document. Link style files enable the navigational component to be easily adjusted, while enforcing a uniformity across related pages, and maintaining a separation of the navigational links from the content. A search engine can generate lists of links to pages of interest, which can be used as a simple kind of personalised tour. Tools are available for analysing documents to help the author develop a suitable document hierarchy and set of keywords. There is also a tool for setting up domain-specific links between pages.

The current version of `curl` (v.2.0) is available as a compressed tar file, called `curl.tar.Z`, by anonymous FTP from `turiel.cs.mu.oz.au` (`128.250.1.10`) in the directory `pub/curl`. Some release specific details can be found on the `curl` Web page:

`http://www.cs.mu.oz.au/~ad/curl/announce.html`

Planned `curl` features for the near future include: support for document printing, guided tours, extended (and faster) search, a user feedback page, and richer navigational links.

More speculative features include: typed links and pages, a cleaner separation of links from the text, dynamic processing to make `curl` more user-centric, and extensions to the 'book' metaphor.

References

Alschuler, L. 1989. "Hand-crafted Hypertext – Lessons from the ACM Experiment", In *The Society of Text.* Edward Barrett (ed.), The MIT Press, pp.342-361.

Bruza, P.D. 1990. "HyperIndicies: A Novel Aid for Searching in Hypermedia" In *ECHT'90: Proc. of the European Conf. on Hypertext.*

Catledge, L.D., and Pitkow, J.E. 1995. "Characterizing Browsing Strategies in the World-Wide Web", *Computer Networks and ISDN Systems*, Vol. 27, pp.1065-1073.

Carlson, P.A. 1989. "Hypertext and Intelligent Interfaces for Text Retrieval", In *The Society of Text.* Edward Barrett (ed.), The MIT Press, pp.58-76.

Conklin, J. 1987. "Hypertext: An Introduction and Survey", *IEEE Computer*, Vol. 20, No. 9, pp17-41.

Edwards, D.M., and Hardman, L. 1989. "Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment", In *Hypertext: Theory into Practice*, Ray McAleese (ed.), Intellect Ltd, Chapter 7, pp.105-125.

Garzotto, F., Mainetti, L., Paolini, P. 1995. "Hypermedia Design, Analysis, and Evaluation Issues", *Comm. of the ACM*, Vol. 38, No. 8, August, pp.74-86.

Garzotto, F., Paolini, P., Schwabe, D., and Bernstein, M. 1991. "Tools for Designing Hyperdocuments", In *Hypertext/Hypermedia Handbook*, Emily Berk and Joseph Devlin (eds.), Chapter 13, McGraw Hill, pp.179-207.

Isakowitz, T., Stohr, E.A., and Balasubramanian, P. 1995. "RMM: A Methodology for Structured Hypermedia Design", *Comm. of the ACM*, Vol. 38, No. 8, August, pp.34-44.

Jones, S., and Cockburn, A. 1996. "A Study of Navigational Support Provided by Two World Wide Web Browsing Applications", In *Hypertext'96: The 7th ACM Conf. on Hypertext*, Washington DC, USA, March, pp.161-169.

McAleese, R. 1989. "Navigation and Browsing in Hypertext", In *Hypertext: Theory into Practice*, Ray McAleese (ed.), Intellect Ltd, Chapter 2, pp.6-44.

Rada, R. 1991. *Hypertext: From Text to Expertext*, McGraw Hill.

Thüring, M., Hannemann, J., and Haake, J.M. 1995. "Hypermedia and Cognition: Designing for Comprehension", *Comms. of the ACM*, Vol. 38, No. 8, August, pp.57-66.

Van Dyke Parunak, H. 1991. "Ordering the Information Graph", In *Hypertext/Hypermedia Handbook*, Emily Berk and Joseph Devlin (eds.), Chapter 20, McGraw Hill, pp.299-325.