



THE GREENSTONE DIGITAL LIBRARY SOFTWARE

Ian H. Witten and Stefan Boddie

*Department of Computer Science
University of Waikato, New Zealand
ihw@cs.waikato.ac.nz, sjboddie@cs.waikato.ac.nz*



The Greenstone Digital Library Software from the New Zealand Digital Library project provides a new way of organizing information and making it available over the Internet. This rapidly developing system has already been adopted by organizations ranging from United Nations agencies to universities and government organizations. Greenstone is open-source software, available from the New Zealand Digital Library (www.nzdl.org) under the terms of the Gnu public license.

Software features

A Greenstone digital library is organized as a set of separate collections. A collection of information comprises several (typically several thousand, or several million) documents, and a uniform interface to these documents. A library may comprise several different collections, each organized in a different way—though there is a strong family resemblance in the way collections are presented.

Accessible via Web browser

Collections are accessed through a standard Web browser (Netscape or Internet Explorer) and combine easy-to-use browsing with powerful search facilities.

Full-text and fielded search

Searching is full-text, and the user can choose between indexes built from different parts of the documents. For example, some collections have an index of full documents, an index of sections, an index of titles, and an

2 THE GREENSTONE DIGITAL LIBRARY SOFTWARE

index of authors, each of which can be searched for particular words or phrases. Results can be ranked by relevance or sorted by a metadata element.

Flexible browsing facilities

Browsing involves hierarchical lists that the user can examine: lists of authors, lists of titles, lists of dates, classification structures, and so on. Different collections may offer different browsing facilities. Indexes for both browsing and searching are constructed during the building process, according to collection configuration information.

Creates access structures automatically

The Greenstone software facilitates maintainability by creating all searching and browsing structures directly from the documents themselves. No links are inserted by hand. This means that if new documents in the same format become available, they can be merged into the collection automatically. Indeed, for many collections this is done by processes that wake up regularly, scout for new material, and rebuild the indexes—all without manual intervention.

Makes use of available metadata

Metadata, which is descriptive information such as author, title, date, keywords, and so on, may be associated with each document, or with individual sections within documents. Metadata is used as the raw material for browsing indexes. It must either be provided explicitly or be derivable automatically from the source documents. The Dublin Core metadata scheme is used for most electronic documents, however, provision is made for other schemes.

Plugins extend the system's capabilities

In order to accommodate different kinds of source documents, the software is organized in such a way that “plugins” can be written for new document types. Plugins currently exist for plain text documents, HTML documents, some proprietary formats, and for recursively traversing directory structures containing such documents. A collection may have source documents in different forms. In order to build browsing indexes from metadata, an analogous scheme of “classifiers” is used: classifiers create browsing indexes of various kinds based on metadata.

Designed for multi-gigabyte collections

Collections can contain millions of documents, making the Greenstone system suitable for collections up to several gigabytes.

Documents can be in any language

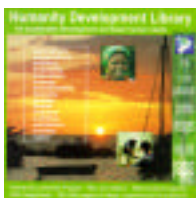
Unicode is used throughout the software, allowing any language to be processed in a consistent manner. To date, collections have been built containing French, Spanish, Maori, Chinese, Arabic and English. On-the-fly conversion is used to convert from Unicode to the user's encoding choice.

THE GREENSTONE DIGITAL LIBRARY SOFTWARE 3

<i>Web interface available in multiple languages</i>	The interface can be presented in multiple languages. Currently, the interface is available in French, Spanish, German, Maori, Chinese, Arabic and English.
<i>Collections can contain text, pictures, audio, and video</i>	Greenstone collections can contain text, pictures, audio and even video clips. Most non-textual material is either linked in to the textual documents or accompanied by textual descriptions (such as figure captions) to allow full-text searching and browsing. However, the architecture permits implementation of plugins and classifiers even for non-textual data.
<i>Uses advanced compression techniques</i>	Compression techniques are used to reduce the size of the indexes and text. Reducing the size of the indexes via compression has the added advantage of increasing the speed of text retrieval.
<i>Collections can be published on the Internet</i>	The software can be used to serve collections over the World-Wide Web. When operating in this mode the software runs under Unix, Windows 95, 98 and NT.
<i>Collections can be published on CD-ROM</i>	Although primarily designed for Internet access over the World-Wide Web, Greenstone collections can be made available, in precisely the same form, on CD-ROM. The user interface is through a standard Web browser (Netscape is provided on each disk), and the interaction is identical to accessing the collection on the Web—except that response times are more predictable. The CD-ROM system works under Windows 3.x, 95, 98, and NT operating systems. The use of compression ensures that the greatest possible volume of information can be packed on to a CD-ROM.

See it in action

The New Zealand Digital Library website (www.nzdl.org) contains numerous example collections, all created with the Greenstone software, that are publicly available for you to peruse. They exemplify various searching and browsing options, and include collections in Arabic, Chinese, French, Maori, and Spanish, as well as English. There are in addition some musical collections. Note that this is a research website and changes occur as we make progress in our work.



The *Humanity Development Library* is a CD-ROM containing 1,230 publications ranging in subject from accounting to water sanitation. It runs under Windows 3.x, 95, 98, and NT, and sample copies can be obtained from greenstone@cs.waikato.ac.nz. The information contained in the CD-ROM can be accessed by searching, browsing by subject, browsing by titles, browsing by organization, browsing a list of how-tos, and by randomly viewing the book covers.

4 THE GREENSTONE DIGITAL LIBRARY SOFTWARE

About this manual

This manual provides a comprehensive introduction to the software and its capabilities. It is divided into four main parts, corresponding to different kinds of use.

The first part (A) gives an overview of the capabilities of the software, and includes a comprehensive glossary of terms used throughout the manual.

The second part (B) is targeted at end users of Greenstone collections. It begins by describing the very simple installation procedure for CD-ROMs. Following that there is a complete account of how to find information in Greenstone collections. In fact, the interface is fairly self-explanatory—the best way to learn is by doing—and this section essentially comprises the on-line help information for a typical collection.

The third part (C) is for those who want to build their own library collections using Greenstone. To do this, you first need some understanding of the files and file structure used within the system, and the overall process of creating a collection—which involves assembling the source documents, “importing” them into the Greenstone system, and then “building” the full-text indexes and browsing structures.

Next we show how to update an existing collection by adding new material and rebuilding it. This is easy to do: the software is designed for ease of maintenance of digital library collections. Building a new collection from scratch can be more challenging. It is easy if there is an existing collection that can be used as an exact model, with the same source file structure and the same kind and format of metadata—for then you can just copy the relevant specifications for the existing collection and rebuild. For genuinely new collections, we describe the “plugins” that are available to accommodate different document formats, and the “classifiers” that are available to build browsing structures—like sorted lists of names or dates, or hierarchical browsers for classification hierarchies. We also describe how you can control the format of what is presented on the screen.

The fourth part (D) describes how to install the full Greenstone software on Unix systems and on Windows 95/98/NT. Most users will not need to do this, and can skip this section. It describes how to put a collection onto a standalone Greenstone CD-ROM that can be distributed for others to use. It gives an account of the directory structure in which the software resides. It is possible to turn on and off user logging, and the log file format is described. There is a maintenance and administration facility

THE GREENSTONE DIGITAL LIBRARY SOFTWARE 5

that allows administrators to examine and control various aspects of a Greenstone installation. Finally we discuss how to translate the interface into other languages that are not yet supported.

The Greenstone software is developing rapidly, and documenting a moving target is difficult. At this stage there are several small things left to do. In some cases these are described here as though they were complete; these sections have been **highlighted**. Most of the highlighting will be removed shortly.



Contents

THE GREENSTONE DIGITAL LIBRARY SOFTWARE	1
Software features	1
See it in action	3
About this manual	4
A—OVERVIEW OF GREENSTONE	6
Glossary	11
B—USING GREENSTONE COLLECTIONS	15
B.1 Using a Greenstone CD-ROM	15
B.2 Finding information in a Greenstone collection	16
How to find information	17
How to read the books	18
What the icons mean	19
How to search for particular words	19
Scope of queries	21
Advanced search features	21
B.3 Changing your preferences	22
Collection preferences	23
Presentation preferences	23
Search preferences	23
C—BUILDING COLLECTIONS	24
C.1 The files in a Greenstone collection	24

Files and directories	25
Creating a collection	26
The imported documents	27
Inside the documents	28
The GML format	29
C.2 Updating existing collections	31
What you have to do	31
Example	34
How it works	34
C.3 Creating new collections	35
The collection configuration file	36
Plugins	40
Classifiers	43
Format strings	47
Examples of classifiers and format strings	48
Subcollections and language-specific indexes	51
D—INSTALLING THE GREENSTONE SOFTWARE	53
D.1 Installing on Unix	54
D.2 Installing on Windows	55
Installing binaries	55
Installing the source code	56
D.3 Making a Greenstone CD-ROM	57
D.4 Configuration files	60
D.5 Where to find the software	61
D.6 User logs	63
D.7 Maintenance and administration	64
User management	64
Information	66
Collections	66
Logs	67
D.8 Translating the interface into other languages	67



A

Overview of Greenstone

A Greenstone digital library is organized as a set of independent collections. A *collection* of information comprises several (typically several thousand, or several million) *documents*, and a uniform interface is provided to all documents in a collection. The collections in a library are each organized in a different way—though there is a strong family resemblance in the way collections are presented.

Collections are accessed through a standard Web browser (for instance, Netscape or Internet Explorer) and combine easy-to-use browsing with powerful search facilities. There are several ways to find information in most Greenstone collections. For example, you can *search for particular words* that appear in the text, or within a section of a document. You can *browse documents by title*: just click on a book to read it. You can *browse documents by subject*. Subjects are represented by bookshelves: just click on a bookshelf to look at the books. Where appropriate, documents come complete with a table of contents: you can click on a chapter or subsection to open it, expand the full table of contents, or expand the full document into your browser window (useful for printing). The international Unicode character set is used internally, so documents can be written in any language. The interface is available in English, French, Spanish, German, Maori, Chinese, and Arabic. The New Zealand Digital Library website (www.nzdl.org) provides numerous example collections.

Along with each collection is a statement of its purpose and coverage, and an explanation of how the collection is organized. Most collections can be accessed by both *searching* and *browsing*. Searching is full-text, and the user can choose between indexes built from different parts of the documents. Some collections have an index of full documents, an index of sections, an index of paragraphs, an index of titles, and an index of section headings, each of which can be searched for particular words or phrases. Browsing involves lists that the user can examine: lists of authors, lists of titles, lists of dates, hierarchical classification structures, and so on. Different collections offer different browsing facilities. Indexes

for both browsing and searching are constructed during a process called *building*, according to instructions in a collection configuration file.

Rich Web browsing facilities can be provided by manually linking parts of documents together and building explicit tables of contents and indexes. However, manually-created structures are difficult to maintain, and inevitably fall into disrepair when the collection expands. The Greenstone software takes a different tack: it facilitates *maintainability* by creating all searching and browsing structures automatically from the documents themselves. No links are inserted by hand. This means that when new documents in the same format become available, they can be merged into the collection automatically. Indeed, for some collections this is done by processes that wake up regularly, scout for new material, and rebuild the indexes—all without manual intervention.

Collections comprise many documents: usually thousands, tens of thousands, or even millions. Each document may be hierarchically organized into *sections* (and subsections, and sub-subsections). Each section comprises one or more *paragraphs*. *Metadata*, which is descriptive information such as author, title, date, keywords, and so on, may be associated with each document, or with individual sections of documents. Metadata is the raw material for browsing indexes. It must either be provided explicitly (for example, in a spreadsheet) or be derivable automatically from the source documents. Metadata is converted to a standard form (using the Dublin Core metadata scheme) and stored with the document for internal use.

In order to accommodate different kinds of source documents, the software is organized in such a way that “plugins” can be written for new document types. Plugins currently exist for plain text documents, HTML documents, some proprietary formats, and for recursively traversing directory structures containing such documents. A collection may have source documents in different forms: it is just a matter of specifying all the necessary plugins. In order to build browsing indexes from metadata, an analogous scheme of “classifiers” is used: classifiers create indexes of various kinds based on metadata. Source documents are brought into the Greenstone system through a process called *importing*, which uses appropriate plugins and classifiers as specified in the collection configuration file.

Greenstone collections can contain text, pictures, audio and even video clips. Compression technology is used to ensure best use of storage. Throughout the Greenstone software, the MG software is used for compression and indexing (see Witten, I.H., Moffat, A. and Bell, T. *Managing Gigabytes: compressing and indexing documents and images*,

10 OVERVIEW OF GREENSTONE

Morgan Kaufmann, second edition, 1999.) Most non-textual material is either linked to the textual documents or accompanied by textual descriptions (such as captions for photos) to allow full-text searching and browsing.

The Greenstone system includes an “administrative” function that enables specified users to examine the composition of all collections, protect documents so that they can only be accessed by registered users on presentation of a password, and so on. Logs of user activity are kept that record all queries made to every Greenstone collection (though this facility can be disabled).

Although primarily designed for Internet access over the World-Wide Web, Greenstone collections can be made available, in precisely the same form, on CD-ROM. A Greenstone CD-ROM can be used on a standalone PC to access the information on the disk. The user interface is through a standard Web browser (Netscape is provided on each CD), and the interaction is identical to accessing the collection on the Web—except that response times are faster and more predictable. If the PC is connected to a network (intranet or Internet), the Greenstone software acts as a network server, using a custom-built Web server provided on each CD, to make exactly the same information available to others who need only use their standard Internet browser software. The software works under all Windows 3.X, 95, 98, and NT operating systems; it can be installed straight off the CD-ROM in a matter of seconds. The use of compression technology ensures that the greatest possible volume of information can be packed on to a CD-ROM.

The collection-serving software operates under Unix (currently tested under Linux and SunOS) and Windows NT, and works with all standard Web servers. A flexible process structure allows **different collections to be served by different computers**, yet be presented to the user in the same way, on the same Web page, as part of the same digital library. Collections can be updated and new ones brought on-line at any time, without bringing the system down; the process responsible for the user interface will notice (through periodic polling) when new collections appear and add them to the list presented to the user. The software architecture allows different parts of a collection to be distributed over several computers, and different collections to be searched together, and a single collection to be replicated on several computers and the user to receive service, transparently, from the one that responds quickest—though these are not yet implemented.

The Greenstone Digital Library Software is open-source software, available from the New Zealand Digital Library (www.nzdl.org) under the

terms of the Gnu public licence. The software includes everything described above: Web serving, CD-ROM creation, collection building, multi-lingual capability, plugins and classifiers for a variety of different source document types. It includes an autoconfigure script to allow easy installation on Unix systems. In the spirit of open-source software, users are encouraged to contribute modifications and enhancements.

Glossary

Term	Meaning
<i>autoconf</i>	Unix program used to configure the Greenstone software installation package to suit your system
<i>Autorun</i>	Windows feature that starts a program automatically whenever a CD-ROM is inserted
Boolean query	Query to an information retrieval system that may contain AND, OR, NOT
Browsing	Accessing a collection by scanning an organized list of metadata values associated with the documents (such as author, title, date, keywords)
<i>buildcol.pl</i>	Program used to build collections
Building	Process of creating the indexing and browsing structures that are used to access a collection
C++	Programming language in which the majority of the Greenstone software is written
Casefolding	Making uppercase and lowercase words look the same, for searching purposes
CGI	Common Gateway Interface, a scheme that allows users to activate programs on the host computer by clicking on Web pages
CGI script	Code associated with a button, menu, or link on a Web page that specifies what the host computer is to do when it is clicked
<i>cgi-bin</i>	Directory in which CGI scripts are stored
Classifier	Greenstone code module that examines document metadata to form an index for browsing
Collection	Set of documents that are brought together under a uniform searching and browsing interface
Collection configuration file	File that specifies how a collection is to be imported and built, what indexes and language interfaces are to be provided, etc
Collection server	Program responsible for providing access to a collection when it is being used
Configuration file	See collection configuration file, main configuration file, site configuration file
CVS	Concurrent Versioning System, a scheme for maintaining source code

12 OVERVIEW OF GREENSTONE

	used throughout Greenstone
<i>db2txt</i>	Tool for viewing a GDBM database as text (see GDBM)
Demo collection	A subset of the Humanities Development Library, distributed with the Greenstone software and used for illustration in this tutorial
Digital library	Collection of digital objects (text, audio, video), along with methods for access and retrieval, and for selection, organization, and maintenance
Document	Basic unit from which digital library collections are constructed; it may include text, graphics, sound, video, etc.
Dublin core	A standard way of describing metadata
Fast CGI	Facility that allows CGI scripts to remain continuously active so that they do not have to be restarted from scratch every time they are invoked
Filter program	That part of a collection server that implements querying and browsing operations
Format string	A string that specifies how documents and other listings are to be displayed
GB-encoding	Standard way of encoding Chinese and other oriental languages
GDBM	Gnu DataBase Manager, a program used within the Greenstone software to store metadata for each document
Gimp	Gnu Image-Manipulation Program used (on Unix) to create icons in Greenstone
GML	Greenstone Markup Language, a file format used for storing documents internally
Gnu license	Software license that permits users to copy and distribute computer programs freely, and modify them—so long as all modifications are made publicly available
Greenstone	The name of this digital library software
GSDL	Abbreviation for Greenstone Digital Library
<i>\$GSDLHOME</i>	Operating system variable that represents the top-level directory in which all Greenstone programs and collections are stored (<i>%GSDLHOME%</i> on Windows systems)
<i>\$GSDLOS</i>	Operating system variable that represents the operating system currently being used (<i>%GSDLOS%</i> on Windows systems)
<i>hashfile</i>	Program used at import or build time to generate the OID of each document
HDL	Humanities Development Library, a Greenstone collection of humanitarian information for developing countries
HTML	HyperText Markup Language, the language in which Web documents are written
<i>import.pl</i>	Program used to import documents

Importing	Process of bringing collections of documents into the Greenstone system
Index	Information structure that is used for searching or browsing a collection
InstallShield	Windows program, used by Greenstone CD-ROMs, that allows a system to be installed from a CD-ROM
Main configuration file	File that contains specifications common to all collections served by this site
Metadata	Descriptive data such as author, title, date, keywords, and so on, that is associated with a document (or document collection)
MG	Managing Gigabytes, a program used by the Greenstone system for full-text indexing, that incorporates compression techniques (see Witten, I.H., Moffat, A. and Bell, T. <i>Managing Gigabytes: compressing and indexing documents and images</i> , Morgan Kaufmann, second edition, 1999)
<i>mgbuild</i>	MG program for building a compressed full-text index
<i>mgquery</i>	MG program for querying a compressed full-text index
<i>mkcol.pl</i>	Program that creates and initializes the directory structure for a new collection
New Zealand Digital Library	Research project in the Computer Science Department at the University of Waikato, New Zealand, that created the Greenstone software (www.nzdl.org)
OID	Object Identifier, a unique identification code associated with a document
Perl	Programming language used for many of the text-processing operations that occur during the building process
Ping	Message sent to a system to determine whether it is running or not
Plugin	Code module for handling documents of different formats, used during the importing and building processes
Protocol	Set of conventions by which a receptionist communicates with a collection server
Ranked query	Natural-language query to an information retrieval system, for which the documents that match the query are sorted in order of relevance
Receptionist	Program that organizes the Greenstone user interface
RTF	Rich Text Format, a standard format for interchange of text documents
Searching	Accessing a collection through a full-text search of its contents (or parts of contents, such as section titles)
Server	See Collection server, Web server
<i>setup.sh</i> , <i>setup.csh</i> , <i>setup.bat</i>	Command used to set up your environment to recognize the Greenstone software
Site configuration file	File that contains specifications used to configure the Greenstone software for the site on which it is installed
Stemming	Stripping endings off a query term to make it more general

14 OVERVIEW OF GREENSTONE

STL	Standard template library, a widely-available library of C++ code
<i>txt2db</i>	Program used at build time to create the GDBM database
Unicode	Standard scheme for representing the character sets used in the world's languages
UNU	The United Nations University; also used to refer to a Greenstone collection created for that organization
Web server	Standard program that computers use to make information accessible over the World Wide Web



B

Using Greenstone collections

The Greenstone software is designed to be easy to use. Web-based and CD-ROM collections have interfaces that are identical. Installing a CD-ROM collection on any Windows computer is very easy indeed; a standard installation setup program is used. A CD-ROM collection can be used locally on the computer where it is installed; also, if this computer is connected to a network, the software automatically and transparently address to all other computers on the network to access the same collection.

If the software is to be used on a Unix server, or if it will be put to serious large-scale use as a Web server, it will be necessary to install the complete Greenstone software and connect it to an existing Web server. This is a more complex procedure because it requires auxiliary software to be installed and may need special configuration options to be set. Most Greenstone users will never have to undertake the installation of the software on their computer; however, for those who do, the procedure is described in section D below.

In Section B.2 we describe the searching and browsing facilities offered by a typical Greenstone collection, the “Demo” collection that is supplied with the Greenstone software. Other collections offer similar facilities; if you can use one, you can use them all.

B.1 Using a Greenstone CD-ROM

To use a Greenstone CD-ROM, just put it into the CD-ROM drive on any Windows PC. Most likely (if “autorun” is enabled on your PC), a window will appear inviting you to install the Greenstone software. If not, find the CD-ROM disk drive (on current Windows systems you can get this by clicking on the *My Computer* icon on the desktop) and double-click it, or the *Setup.exe* file inside it. In either case the Greenstone *Setup* program will be entered, which guides you through the setup procedure. Most people respond *yes* to all the questions except for the one which offers to

16 USING GREENSTONE COLLECTIONS



Figure 1 Using the Demo collection

install the Netscape browser; if you already have a browser you probably don't need to install a new one.

When the installation procedure has finished, you'll find the library in the *Programs* submenu of the Windows *Start* menu, under the name of the collection (for example, "Humanity Libraries" or "United Nations University"). There will be a *Standalone Version* as well as a *Server or Network* version, and an *Uninstall* option which will

completely remove the library and associated software from your system should you so desire. You should use the *Server or Network* version whether or not you are connected to a network; the *Standalone Version* is only supplied in case the system experiences difficulty in ascertaining your network setup.

Once the software has been installed, the library will be entered automatically every time you re-insert the CD-ROM if autorun is enabled.

B.2 Finding information in a Greenstone collection

The easiest way to learn to use a Greenstone collection is to try it out. Don't worry—you can't break anything. Click liberally: most images that appear on the screen are clickable. If you hold the mouse stationary over an image, most browsers will soon pop up a message that tells you what will happen if you click.

Experiment! Choose common words like "the" and "and" to search for—that should evoke some hits, and nothing will break.

Greenstone digital library systems often comprise several separate collections—for example, computer science technical reports, literary works, internet FAQs, magazines. There will be a home page for the digital library system which allows you to access any collection; in addition, each collection has its own "about" page that gives you information about how the collection is organized and the principles

B.2 FINDING INFORMATION IN A GREENSTONE COLLECTIONS 17

governing what is included in it. To get back to the “about” page at any time, just click on the “collection” icon that appears at the top left side of all searching and browsing pages.

Figure 1 shows a screenshot of the “Demo” collection supplied with the Greenstone software, which is a very small subset of the Humanity Development Library collection; we will use it as an example to describe the different ways of finding information. (If you can’t find the Demo collection, use the Humanity Development Library instead; it looks just the same.) First, almost all icons are clickable. Several icons appear at the top of almost every page; Table 1 shows you what they mean.

Table 1 What the icons at the top of each page mean

demo	This takes you to the “about” page
HOME	This takes you to the Digital Library’s home page, from which you can select another collection
HELP	This provides help text similar to what you are reading now
PREFERENCES	This allows you to set some user interface and searching options that will then be used henceforth

The “*search ... subjects ... titles a-z ... organization ... how to*” bar underneath gives access to the searching and browsing facilities. The leftmost button is for searching, and the ones to the right of it—four, in this collection—evoke different browsing facilities. These may differ from one collection to another.

How to find information

Table 2 shows the five ways to find information in the Demo collection.

You can *search for particular words* that appear in the text from the “search” page. (This is just like the “about” page shown in Figure 1, except that it doesn’t contain the *about this collection* text.) The search page can be reached from other pages by pressing the *search* button. You can *access publications by subject* by pressing the *subjects* button. This brings up a list of subjects, represented by bookshelves. You can *access publications by title* by pressing the *titles a-z* button. This brings up a list of books in alphabetic order. You can *access publications by organization* by pressing the *organization* button. This brings up a list of organizations. You can *access publications by how to listing* by pressing the *how to* button. This brings up a list of “how to” hints. You can see these buttons in Figure 1.

18 USING GREENSTONE COLLECTIONS

Table 2 What the icons on the search/browse bar mean

search	Search for particular words
subjects	Access publications by subject
titles a-z	Access publications by title
organization	Access publications by organization
how to	Access publications by “how to” listing

How to read the books

In the Demo collection, you can tell when you have arrived at an individual book because there is a photograph of its front cover (Figure 2). Beside the photograph is a table of contents with an arrow marking where you are. This table is expandable: click on the folders to open them or close them. Click on the open book at the top to close it.

Underneath is the text of the current section (*Introduction and Summary* in the example, beginning at the very bottom of the illustration). When you have read through it, there are arrows at the end to take you on to the next section or back to the previous one.

Below the photograph are four buttons. Click on *detach* to make a new browser window for this book. (This is useful if you want to compare

books, or read two at once.) If you have reached this book through a search, the search terms will be highlighted: the *no highlighting* button turns this off. Click on *expand text* to expand out the whole text of the current section, or book. Click on *expand contents* to expand out the whole table of contents so that you can see the titles of all chapters and subsections.

In some collections, the documents do not have this kind of hierarchical structure. In this case, no table of contents is

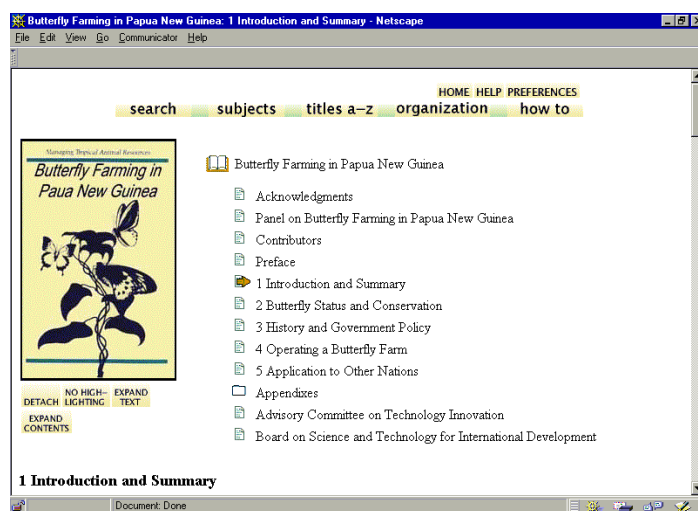


Figure 2 A book in the Demo collection













B.2 FINDING INFORMATION IN A GREENSTONE COLLECTIONS 19

displayed when you get to an individual document—just the document text. In some cases, the document is split into pages, and you can read sequentially or jump about from one page to another.

What the icons mean

When you are browsing around the collection, you will encounter the items shown in Table 3.

Table 3 Icons that you will encounter when browsing

	Click on a book icon to read the corresponding book
	Click on a bookshelf icon to look at books on that subject
	View this section of the text
	Open this folder and view contents
	Click on this icon to close the book
	Click on this icon to close the folder
	Click on the arrow to go on to the next section ...
	... or back to the previous section
	Open this page in a new window
	Expand table of contents
	Display all text
	Highlight search terms

How to search for particular words

From the search page, follow these simple steps to make a query:

- Specify what items you want to search: in the Demo collection you can search books, chapters, or section titles.
- Say whether you want to search for all or just some of the words
- Type in the words you want to search for into the query box
- Click the *Begin Search* button

When you make a query, the titles of up to twenty matching documents will be shown. There is a button at the end to take you on to the next twenty. From there you will find buttons to take you on to the third twenty or back to the first twenty, and so on. However, for efficiency

20 USING GREENSTONE COLLECTIONS

reasons a maximum of 100 is imposed on the number of documents returned. You can change these numbers by clicking the *preferences* button at the top of the page.

Click the title of any document, or the little icon beside it, to open it. The icon may show a book, or a folder, or a page: it will be a book icon if you are searching books; otherwise if you are searching sections it will be a folder or page icon depending on whether or not the section found has subsections.

SEARCH TERMS

Whatever you type into the query box is interpreted as a list of words called “search terms.” Each search term contains nothing but alphabetic characters and digits. Terms are separated by white space. If any other characters such as punctuation appear, they serve to separate terms just as though they were spaces. And then they are ignored. You can’t search for words that include punctuation.

For example, the query

```
Agro-forestry in the Pacific Islands: Systems for  
Sustainability (1993)
```

will be treated the same as

```
Agro forestry in the Pacific Islands Systems for  
Sustainability 1993
```

QUERY TYPE

There are two different kinds of query.

- Queries for all the words. These look for documents (or chapters, or titles) that contain all the words you have specified. Books that satisfy the query are displayed, **in alphabetical order**.
- Queries for some of the words. Just list some terms that are likely to appear in the documents you are looking for. Books are displayed in order of how closely they match the query. When determining the degree of match,
 - the more search terms a document contains, the closer it matches;
 - rare terms are more important than common ones;
 - short documents match better than long ones.

Use as many search terms as you like—a whole sentence, or even a whole paragraph. If you specify only one term, it doesn’t much matter whether you use an *all* or a *some* query, except that in the second case documents

B.2 FINDING INFORMATION IN A GREENSTONE COLLECTIONS 21

will be ordered by the search term's frequency of occurrence.

Scope of queries

In most collections you can choose different indexes to search. For example, there might be author or title indexes. Or there might be chapter or paragraph indexes. Generally, the full matching document is returned regardless of which index you search.

If documents are books, they will be opened at the appropriate place.

Advanced search features

While the above is enough to meet most searching needs, some more advanced search features are provided. These are activated from the *preferences* page, which is reached by clicking the *preferences* button at the top of the page—see section B.3 below. When you change your *preferences*, do not click your browser's *Back* button—that would undo the changes. Instead, click any of the buttons on the search/browse bar.

CASE SENSITIVITY AND STEMMING

When you specify search terms, you can choose whether upper and lower case must match between the query and the document: this is called “case sensitivity.” You can also choose whether to ignore word endings or not: this is called “stemming.”

Under *Search options* on the *Preferences* page you will see a pair of buttons labeled *ignore case differences* and *upper/lower case must match*; these control the case sensitivity of your queries. Below is a pair of buttons labeled *ignore word endings* and *whole word must match*: these control stemming.

For example, if the buttons *ignore case differences* and *ignore word endings* are selected, the query

African building

will be treated the same as

africa builds

because the uppercase letter in “African” will be transformed to lowercase, and the suffixes “n” and “ing” will be removed from “African” and “building” respectively (also, “s” would be removed from “builds”).

22 USING GREENSTONE COLLECTIONS

Generally case differences and word endings should be ignored unless you are querying for particular names or acronyms.

PHRASE SEARCHING

If your query includes a phrase in quotation marks (“ and ”), only documents containing that phrase, exactly as typed, will be returned.

If you want to use phrase searching, you need to learn a little about how it works. Phrases are processed by a post-retrieval scan. First the query is issued in the normal way—all the words in the phrase are included as search terms—and then the documents returned are scanned to eliminate those in which that phrase does not appear.

During the post-retrieval scan, phrases are checked just as they are, including any punctuation. For example, the query

```
what's a "post-retrieval scan?"
```

will first retrieve all documents that match all of the words

```
what s a post retrieval scan
```

and then the documents returned will be checked for the phrase

```
post-retrieval scan?
```

Finally, for computational reasons, the smallest unit of granularity is automatically enforced whenever phrase searching is used. In most collections, this is the paragraph level.

ADVANCED QUERY MODE

In *advanced query mode*, which can be selected on the *Preferences* page, the queries for *all* of the words, described above, are actually Boolean queries. They consist of a list of terms joined by logical operators & (and), | (or), and ! (not). Absent operators are interpreted as & (and): thus a query without any operators returns documents that match *all* the terms.

If the words AND, OR, and NOT appear in your query they are treated as ordinary search terms, not operators. For operators you must use &, |, and !. In addition, parentheses can be used for grouping.

B.3 Changing your preferences

When you click the *preferences* button at the top of the page you will be

able to change some features of the interface to suit your own requirements.

Collection preferences

Some collections comprise several subcollections, which can be searched independently or together, as one unit. If so, you can select which subcollections to include in your searches on the Preferences page.

Presentation preferences

Depending on the particular collection, there may be several options you can set that control the presentation. Collections of Web pages allow you to suppress the Greenstone navigation bar at the top of each document page, so that once you have done a search you land at the exact Web page that matches without any Greenstone header. To do another search you will have to use your browser's "back" button. These collections also allow you to suppress Greenstone's warning message when you click a link that takes you out of the digital library collection and on to the Web itself. And in some Web collections you can control whether the links on the "Search Results" page take you straight to the actual URL in question, rather than to the digital library's copy of the page. Collections that are capable of being presented in different languages allow you to specify the interface language. If the language is Chinese, you can also specify which of the standard Chinese encodings your browser uses. Finally, all collections allow you to switch to a textual interface format rather than the standard graphical one. This is particularly useful for visually impaired users who use large screen fonts or speech synthesizers for output.

Search preferences

Two pairs of buttons control the case sensitivity and stemming of the searches that you make. The first set of buttons controls whether upper and lower case must match (case sensitivity). The second set controls whether to ignore word endings or not (stemming).

You can also switch to an "advanced" query mode which allows you to combine terms using AND (&), OR (|), and NOT (!). This allows you to specify more precise queries.

Finally, you can control the number of hits returned, and the number presented on each screenful.



C

Building collections

We now look at the process of building collections. This is for librarians!—but Greenstone empowers anyone to become a digital librarian. The next section describes the computer files that are used in a collection, including the original source files that contain the source material, and the file structure that is used internally. Each document can be hierarchically structured, with sections and subsections.

Section C.2 describes how to update an existing collection by adding new material and rebuilding it. This is easy to do: the software is designed for ease of maintenance of digital library collections.

Then we look at the more challenging process of building a new collection from scratch. Building simple collections is easy—to build a collection with exactly the same structure as an existing one, with source documents in exactly the same format, is extremely simple. The Greenstone software is very flexible, allowing different kinds of source document and different sorts of browsing indexes, and this flexibility comes at a price: if you want to take advantage of it, you must learn to control it. Section C.3 describes all the issues that are involved in building new collections.

C.1 The files in a Greenstone collection

When you create a new collection or add material to an existing one, you need to put the original source documents in a place from which they can be brought into the system, a process known as “importing” the documents into Greenstone. When they are imported, documents are converted into a simple HTML-like format known as GML for Greenstone Markup Language, which includes any “metadata,” or descriptive data such as author, title, date, keywords, and so on, that is associated with the document. Documents are assumed to be in Unicode, expressed in the UTF-8 coding scheme (of which the ASCII characters form a subset).

Files and directories

The Greenstone system is placed in a directory structure rooted at *\$GSDLHOME* (on the New Zealand Digital Library project's computers, it's rooted in */home/nzdl/gsd1*). In order to use the software, you need to set this environment variable accordingly.

There are some other initialization functions that need to be done in order to make best use of the software; for example, augmenting your search path so that Greenstone utility programs are found automatically. These functions can be accomplished automatically by a setup script in *\$GSDLHOME/setup.sh* (or *setup.csh* if you are using the C-shell; for other shells consult your local Unix expert). On Windows, the directory structure is rooted at *%GSDLHOME%*, and the setup script is in the file *%GSDLHOME%\setup.bat*. Table 4 gives Unix and Windows experts more information on what is required.

We will generally omit the *\$GSDLHOME* prefix from filenames in this documentation.

Table 4 Contents of the file *setup.sh*

<i>\$GSDLHOME/setup.sh</i>	<pre>export GSDLHOME=/home/nzdl/gsd1 export GSDLOS=`uname -s tr A-Z a-z` export PATH=\$PATH:\$GSDLHOME/bin/script:\$GSDLHOME/bin/\$GSDLOS</pre>	<i>set by auto</i>
<i>%GSDLHOME%\setup.bat</i>	<pre>set GSDLHOME=c:\gsdl set GSDLOS=windows set PATH=%PATH%;%GSDLHOME%\bin\script;%GSDLHOME%\bin\windows;</pre>	

The directory *collect* (i.e. *\$GSDLHOME/collect*) contains a directory for each collection; to see what collections there are, both public and private, you can look here. Collection names usually bear some resemblance to what the collection is called on the New Zealand Digital Library Web page, although that page, of course, only shows public collections. For each collection, there will be several subdirectories, shown in Table 5a. The contents of these directories for the Demo collection are show in Table 5b, for reference. The *perllib* directory contains PERL programs that are specific to this collection.

26 BUILDING COLLECTIONS

Table 5 (a) Subdirectories for an individual collection ; (b) Subdirectories for the Demo collection

(a)	<i>archives</i>	Source files from which the collection is built, usually in GML format
	<i>building</i>	Stores the collection's indexes while the collection is being built
	<i>etc</i>	Collection-specific configuration file <i>collect.cfg</i> , and any necessary data files
	<i>import</i>	Original raw material for the collection, in its original format
	<i>index</i>	Stores the indexes for the live collection, as it is served to users
	<i>perllib</i>	Contains PERL programs specific to this collection
(b)	<i>collect/demo</i>	<pre> archives: HASH0105.dir HASH017d.dir HASH63e6.dir HASHaad6.dir HASH0144.dir HASH026b.dir HASH7df3.dir HASHe52a.dir HASH0173.dir HASH54cf.dir HASHa0a5.dir archives.inf building: etc: collect.cfg mags.txt sub.txt org.txt import: bostid ecourier faobetf index.txt wb index: build.cfg dtx stt stx text perllib: classify </pre>

Creating a collection

The process of creating a collection goes like this. The original raw material is placed in the *import* directory, in whatever form it is available—plain text files, Word documents, HTML files, depending on where it came from. Then the *import* process is invoked. Controlled by the configuration file in *etc/collect.cfg*, the files in *import* are converted into GML format and placed in the *archives* directory. The original raw material remains in *import*. Then the *build* process is invoked. Again controlled by the configuration file, the requisite indexes for the collection are built and placed in the *building* directory. Finally, the contents of the *building* directory are moved into the *index* directory, and the new version of the collection automatically becomes live.

This procedure may seem cumbersome. But all the steps are necessary for efficient operation with large collections. First, the *import* process could be performed on the fly during the building operation—but because building indexes is a multipass operation, the often lengthy importing would be repeated several times. Second, the *build* process can take

C.1 THE FILES IN THE GREENSTONE COLLECTION 27

considerable time—some days, for very large collections. It puts the result into the separate *building* directory so that, if the collection already exists, it can continue to be served to users in its old form from the directory *index* throughout the building operation.

If desired, the separate step of importing can be avoided by putting the original raw material in the *archives* directory. The building process uses filename extensions to recognize the type of files being dealt with. On finding files other than *.gml* files in the archives directory, it will invoke the import procedure automatically for each pass of index creation. This saves the extra space required to store the collection twice, in both original and imported form, at the expense of the additional time required to perform the import conversion for each pass of the building process.

The configuration file that controls the collection creation procedure resides in the collection's *etc* directory, along with any collection-specific data files that are required. (For example, building a hierarchy of classifications requires a data file of sub-classifications, which is placed here.)

The imported documents

In order to identify documents in the Greenstone system, a unique object identifier or “OID” is assigned to each original source document and stored as metadata within that document. It is important that no two documents have the same OID, and that a document's OID persists throughout the index-building process—so that a user's search history is unaffected by rebuilding the collection. OIDs are assigned by hashing the contents of the original source document during the *import* process; an example OID is “HASHa723e7e164df07c833bfc4”.

Once it has been imported, each document is stored in its own subdirectory of the *archives* directory, along with any other files—for example, image files—that are associated with the document. The name of the subdirectory is generated from the document's OID, and the document is stored in GML format in a file called *doc.gml*.

To retain compatibility with Windows 3.X systems, only eight characters are used in directory and file names. The much longer OIDs are mapped into multiple directory names, but the directory structure is kept as shallow as possible by using only as much of the OID as required. For example, just the first eight characters of the documents OID are used, except that if a directory with this name already exists, a subdirectory of it is created using the next eight, and so on (in fact, this seldom happens).

28 BUILDING COLLECTIONS

Inside the documents

Within a single document, the GML format imposes a limited amount of structure. (We will give examples shortly, in Table 8.) Documents are divided into paragraphs. They can be split hierarchically into sections and subsections; these may be nested to any depth. OIDs are extended to identify sections and subsections by appending section and subsection numbers, separated by periods, to a document's OID. For example, subsection 3 of section 2 of document HASHa7 is referred to as HASHa7.2.3. When you read a book in a Greenstone collection, the section hierarchy is manifested in the table of contents of the book. For example, books in the Demo collection have a hierarchical table of contents showing chapters, sections, and subsections. Documents in the Computer Science Technical Reports do not have a hierarchical subsection structure, but each document is split into pages and you can browse around the pages of a retrieved document. Chapters, sections, subsections, and pages are all implemented simply as "sections" within the document.

The document structure is also used for searchable indexes. There are three levels of index: *documents*, *sections*, and *paragraphs*. A document index contains the full document. When a section index is created, each section stretches from a *gsdlsection* tag to the next-occurring *gsdlsection* tag—thus a chapter that immediately begins with a new section will produce an empty document in the index. Sections and subsections are treated alike: the hierarchical document structure is flattened for the purposes of creating searchable indexes. As well as indexes of text, indexes of any kind of metadata can also be created. For example, most collections offer searchable indexes of section titles.

It has already been mentioned that metadata is stored with documents. The Dublin Core metadata standard is used, which defines the metadata types in Table 6 (starred ones are used in current collections). However, metadata types that are not in the Dublin Core may be used too (for example, the Demo collection contains "how to" and "Magazine" metadata).

Table 6 The Dublin Core metadata standard. Starred items are used in current collections

Name	Metadata subtag	Definition
*Title	Title	A name given to the resource
*Creator	Creator	An entity primarily responsible for making the content of the resource
*Subject and keywords	Subject	The topic of the content of the resource
*Description	Description	An account of the content of the resource
*Publisher	Publisher	An entity responsible for making the resource available
Contributor	Contributor	An entity responsible for making contributions to the content of the resource
*Date	Date	A date associated with an event in the life cycle of the resource
Resource type	Type	The nature or genre of the content of the resource
Format	Format	The physical or digital manifestation of the resource
*Resource identifier	Identifier	An unambiguous reference to the resource within a given context: this is the object identifier or OID
*Source	Source	A Reference to a resource from which the present resource is derived
*Language	Language	A language of the intellectual content of the resource
Relation	Relation	A reference to a related resource
Coverage	Coverage	The extent or scope of the content of the resource
Rights management	Rights	Information about rights held in and over the resource

The GML format

All text documents are converted to GML when they are imported into the Greenstone system. The HTML convention of tags enclosed in angle brackets is adopted for markup; any <, >, or " characters within the text are stored as their HTML equivalent (< > and ").

A <gSDLsection> tag denotes the start of each document section, and the corresponding </gSDLsection> closing tag marks the end of that section. Each <gSDLsection> tag can contain any number of subtags shown in Table 7. Metadata is indicated within the gSDLsection tag (for example, <gSDLsection Title="Food and Nutrition Bulletin">); thus different metadata can be associated with individual sections of a document. Any subtags other than those in are considered to be metadata that is attached to that section.

30 BUILDING COLLECTIONS

Table 7 Subtags of <gsdlsection>

gsdlsourcefilename	Original file from which the GML was generated
gsdldoctype	Type of document (currently the only recognized type is indexed_doc)
gsdlassocfile	File associated with the document (e.g. an image file)
gsdlnum	Subsection number (this does not exist at the document's top level)

Table 8a is a GML file that contains a simple document comprising a single section with title, and three associated images. Table 8b shows a more complex document: a book with two sections called *Preface* and *Conclusions*, the second of which has two subsections. Note that a chapter is simply treated as a top-level section. In some collections documents are split into individual pages. These are treated as sections, though they don't usually have titles. Table 8c shows a book with two sections (corresponding to chapters), the first of which has two pages and the second one page.

Table 8 Three examples of documents in GML format

- (a)

```
<gsdlsection
  gsdlsourcefilename = "uu02fe.txt"
  gsdldoctype = "indexed_doc"
  Identifier="HASHa723e7e164df07c833bfc4"
  Title = "Freshwater Resources in Arid Lands"
  gsdlassocfile="cover.jpg:image/jpeg"
  gsdlassocfile="p21.jpg:image/jpeg"
  gsdlassocfile="p22.jpg:image/jpeg">
  This is the text of the document
</gsdlsection>
```
-
- (b)

```
<gsdlsection
  gsdlsourcefilename = "uu02fe.txt"
  gsdldoctype = "indexed_doc"
  Identifier="HASHa723e7e164df07c833bfc4"
  Title = "Freshwater Resources in Arid Lands">
  <gsdlsection gsdlnum="1" Title="Preface">
    This is the text of the preface
  </gsdlsection>
  <gsdlsection gsdlnum="2" Title="Conclusions">
    <gsdlsection gsdlnum="1" Title="Part 1">
      This is the first part of the conclusions
    </gsdlsection>
    <gsdlsection gsdlnum="2" Title="Part 2">
      This is the second part of the conclusions
    </gsdlsection>
  </gsdlsection>
</gsdlsection>
```
-

```
(c) <gsdlsection
  gsdlsourcefilename = "uu02fe.txt"
  gsdldoctype = "indexed_doc"
  Identifier="HASHa723e7e164df07c833bfc4"
  Title = "Freshwater Resources in Arid Lands">
  <gsdlsection gsdlnum="1" Title="Chapter 1">
    <gsdlsection gsdlnum="1">
      This is the text of the first page of chapter 1
    </gsdlsection>
    <gsdlsection gsdlnum="2">
      This is the text of the second page of chapter 1
    </gsdlsection>
  </gsdlsection>
  <gsdlsection gsdlnum="2" Title="Chapter 2">
    <gsdlsection gsdlnum="1">
      This is the text of the first and only page of chapter 2
    </gsdlsection>
  </gsdlsection>
</gsdlsection>
```

C.2 Updating existing collections

Updating an existing collection is easy. Since the collection already exists, its directory (a subdirectory of *collect*) will exist, with subdirectories *import*, *archives*, *building*, *index*, and *etc*. The *etc* directory will already contain a collection configuration file *collect.cfg*.

The collection is to be updated with new files. For example, in the Demo collection the raw material consists of marked up HTML files containing <<TOC>> tags to split books into sections and subsections, and <<I>> tags to indicate where an image is to be inserted. For each book in the library there is a directory that contains a single HTML file representing the book, and separate files containing the associated images. We assume that the new material is in exactly the same format as the existing files that have previously been used to create the collection.

What you have to do

You need to import the new material into the Greenstone system, and then rebuild the collection. The procedure is as follows. Put the new material into the *import* directory, and then import it by executing *import.pl*:

```
import.pl [options] collection-name
```

The program *import.pl* processes the files and directories in the *import* directory and converts the documents to GML format, placing the result in the *archives* directory. Any other files (e.g. image files) that are associated with a document are also copied to *archives*.

32 BUILDING COLLECTIONS

The options for *import.pl* are given in the table below. You can override the default directory names *import* and *archives*, and specify that just a few documents are to be converted—which is useful for testing purposes when dealing with large collections. To add new material to a collection without re-importing the old, put the new material into another directory and specify it using the *—importdir* option.

The purpose of the *import* step is to put all documents into a standard format before building the collection. Otherwise each document would have to be converted several times, because building is a multi-pass operation. However, it is possible to simply omit this step. In this case the original source material is put straight into the *archives* directory, and the program *import.pl* is not executed at all. The building step processes all files in the *archives* directory. Noticing from the filename extensions that they need to be converted first, the building process silently invokes this step for each pass.

Once the new material has been imported, the collection should be rebuilt:

```
buildcol.pl [options] collection-name
```

The final step is to remove the old contents of the *index* directory, if any exist, and move the contents of the *building* directory into the *index* directory, from which the collection is served to users. On Unix systems,

```
rm -r index/*
mv building/* index
```

The *building* directory is merely used as a holding-place during the potentially lengthy building process. It is quite possible that there are active users of the collection at the time that *building* is moved into *index*. Since documents are referred to internally by their OID, which persists through the rebuilding process, users who are examining the results of a query or browse operation will still access the expected documents. If a query is actually in progress at the time the contents of *index* are changed, it may encounter an internal MG error which will cause it to be transparently re-executed, this time on the new version of the collection.

Some useful debugging features are incorporated into *import.pl* and *buildcol.pl*. You can perform a *build* operation but print (to *stdout*) the information that is generated instead of passing it to MG or *txt2gdbm*. Suppose, for example, you are writing a plugin to handle a particular collection, and the Greenstone interface shows that the text is coming out wrong. Running *buildcol.pl* with the *debug* switch and the appropriate *mode* option will show exactly what the plugin is sending to MG

In summary, once you have put the raw material into the *import* directory these four commands are all you need to rebuild a collection:

```
import.pl collection-name
buildcol.pl collection-name
rm -r index/*
mv building/* index
```

Table 9 Options for *import.pl*, *buildcol.pl* and *mkcol.pl*

	Option	Meaning
<i>import.pl</i>	-verbosity <number>	0=none, 3=too much
	-importdir <directory>	Where the original material comes from (default <i>import</i>)
	-archivedir <directory>	Where the converted material goes to (default <i>archives</i>)
	-cachedir <directory>	Not yet implemented
	-maxdocs <number>	Maximum number of documents to convert
	-removeold	Delete everything in the <i>archives</i> directory first
	-gzip	Compress the resulting GML files using <i>gzip</i>
	-debug	Print out the GML documents instead of saving them to GML files
<i>buildcol.pl</i>	-verbosity <number>	0=none, 3=too much
	-archivedir <directory>	Where the building material comes from (default <i>archives</i>)
	-builddir <directory>	Where the indexes that are built go to (default <i>building</i>)
	-cachedir <directory>	Cache the archives to this directory (used to specify a local disk when archives are on a remote disk)
	-maxdocs <number>	Maximum number of documents to include
	-allclassifications	Normally, information for empty classifications created by a classifier is suppressed. If this flag is set, it is included (this may create empty folders).
	-debug	Print the output that would normally be piped to MG (when compressing text or building an index) or <i>text2db</i> (when writing the GDBM file)
	-mode all	Just compress the text, or just build the index(es), or just write the GDBM database (<i>all</i> , the default, is a full build)—for use in conjunction with
	-mode compress_text	-debug
	-mode build_index	
	-mode infodb	
	-keepold	Do not delete the current contents of the <i>building</i> directory before starting (defaults to <i>false</i>)—for use in conjunction with <i>-mode</i> to continue after a failure of the build process

34 BUILDING COLLECTIONS

	-index	If not set, all indexes in <i>collect.cfg</i> are built; this switch allows selective indexing for debugging (e.g. <code>-index document:Title</code>)
<i>mkcol.pl</i>	-maintainer	Email address of the collection's maintainer
	-public	Whether collection is to be made public or not
	-beta	Whether collection is beta version or not
	-index	List of indexes to include
	-defaultindex	The default index
	-title	Title of the collection
	-about	"About this collection" text

Example

Consider as an example the Demo collection. There are two ways to build the collection. The first is to install the raw material in the directory *collect/demo/import*, run *import.pl* to convert to GML and copy to the *archives* directory, then run *buildcol.pl* on the resulting archives. The second is to install the raw material in the directory *collect/demo/archives* and run *buildcol.pl* directly on this raw material.

The first method is much faster than the second, because the conversion process is done once only, instead of at each pass of the build operation (there are two passes to compress the text, two for each index, and a further pass to construct the GDBM file). However, the second method uses less disk space, because no GML files are stored, and retains the original directory structure.

Once the collection has been built, it will be discovered automatically by the receptionist, which polls the *collect* directory regularly to update its list of collections.

How it works

The original material in the *import* directory may be in any format, and a "plugin" is required to process each format type (described below). The plugins that a collection uses must be specified in the collection configuration file. The *import* program reads the list of plugins and passes each document to each plugin in order until it finds one that can process it. Since we are working with an existing collection, we have assumed that all plugins necessary to process new material have already been specified in the configuration file.

The *buildcol.pl* step actually creates the indexes for both searching and

browsing. The MG software is generally used to do the searching. The *buildcol.pl* process invokes the MG module *mgbuild* to create each of the indexes that is required. For example, the Humanities Development Library has three indexes, one for entire books, one for chapters, and one for section titles. (Subdirectories of the *index* directory are created for each of these indexes.)

MG also compresses the text of the collection (the compressed version is placed in a subdirectory *index/text*). In addition, the image files are linked into the *index/assoc* subdirectory. Now none of the material in the *import* and *archives* directories is needed to run the collection (though they would be needed to rebuild it).

Associated with each collection is a database stored in GDBM (Gnu database manager) format. This contains an entry for every section of every document giving its OID, its internal document number used by MG (which, unlike the OID, may change from one build to the next), and metadata such as its title. Information for each of the browsing indexes, which appear as buttons on the Greenstone search/browse bar, is also extracted during the building process and stored in the GDBM database. A “classifier” program (see below) is required for each browsing index to extract the appropriate information from GML documents. Like plugins, classifiers are written on an *ad hoc* basis for the particular information required, and where possible are reused from one collection to another.

The program *buildcol.pl* builds the indexes based on whatever appears in the *archives* directory. The first plugin specified by all collections is one that processes GML files (called *GMLPlug*), and so if *archives* contains files created by *import.pl* they will be processed correctly. If the *archives* directory contains material in the original format, it will be converted using the appropriate plugin for that format. This is why the *import* process is an optional one (although usually desirable because it speeds up the operation substantially).

C.3 Creating new collections

The first step in creating a new collection is to run the program *mkcol.pl*.

```
mkcol.pl [options] collection-name
```

This sets up the directory structure and generates a skeleton collection configuration file. The options are shown below. Only the creator is required, all others have defaults. For example, you might type

```
mkcol.pl -creator ... collection-name
```

36 BUILDING COLLECTIONS

The original source material should be placed in the *import* directory. The next step is to run *import.pl* to import the documents into the *archives* directory, and then *buildcol.pl* to create the indexes and compressed text in the *building* directory as described above. Both these operations are controlled by the collection's configuration file, *etc/collect.cfg*. You will need to edit this file to specify appropriate plugins to decipher the imported file format and convert it to the GML language; without these, the collection that is built may be empty because files for which there is no appropriate plugin are simply ignored (and a warning message is printed to the error channel). You will need to edit the configuration file to specify which full-text search indexes are to be built: the default is a single index for the entire text of the collection. You will need to specify appropriate classifiers to create the information necessary for browsing interfaces. In the absence of these specifications, the configuration file created by *mkcol.pl* will make a bland collection with just one searchable index and no browsing indexes—and it may be empty because no documents can be imported.

Thus the entire sequence needed to create a new collection is

```
mkcol.pl -creator ... collection-name
           place the source material in the import directory
           edit the configuration file etc/collect.cfg appropriately
import.pl collection-name
buildcol.pl collection-name
mv building/* index
```

The collection configuration file

All you have to do now is to come up with an appropriate configuration file and re-run the import and build processes. Look at the one you have in *etc/collect.cfg*: you will find that it looks something like Table 10a. Lines are numbered for reference: they can appear in any order, each line being an independent command chosen from the list in Table 10c followed by a number of fields separated by white space.

Table 10 (a) Plain collection configuration file, as created by *mkcol.pl*; (b) Collection configuration file for the Demo collection; (c) Summary of configuration file commands

(a)	creator	sjboddie@cs.waikato.ac.nz	a1
	maintainer	sjboddie@cs.waikato.ac.nz	a2
	public	true	a3
	beta	true	a4
	indexes	document:text	a5
	defaultindex	document:text	a6
	plugins	GMLPlug	a7

C.3 CREATING NEW COLLECTIONS 37

	plugins	TEXTPlug	a8
	plugins	ArcPlug	a9
	plugins	RecPlug	a10
	classify	AZList metadata=Title	a11
	collectionmeta	collectionname "my_new_collection"	a12
	collectionmeta	.document:text "documents"	a13
<hr/>			
(b)	creator	sjboddie@cs.waikato.ac.nz	b1
	maintainer	sjboddie@cs.waikato.ac.nz	b2
	public	true	b3
	beta	true	b4
	indexes	section:text section:Title document:text	b5
	defaultindex	section:text	b6
	plugin	GMLPlug	b7
	plugin	HBPlug	b8
	plugin	ArcPlug	b9
	plugin	IndexPlug	b10
	plugin	RecPlug	b11
	classify	Hierarchy hfile=sub.txt metadata=Subject sort=Title	b12
	classify	HDLList Title	b13
	classify	Hierarchy hfile=org.txt metadata=Organization sort=Title	b14
	classify	List metadata=Howto	b15
	format	SearchVlist "<td valign=top>[link][icon][link]</td> <td>{If}{[parent(All': '):Title],[parent(All': '):Title]: } [link][Title][link]</td>"	b16
	format	CL4Vlist " [link][Howto][link]"	b17
	format	DocumentImages true	b18
	format	DocumentText "<h3>[Title]</h3>\n\n<p>[Text]"	b19
	collectionmeta	collectionname "greenstone demo"	b20
	collectionmeta	collectionextra "This is a demonstration collection for the Greenstone digital library software.\nIt contains a small subset (11 books) of the Humanity Development Library"	b21
	collectionmeta	iconcollection "http://www.nzdl.org/gsd/collect/demo/images/demo.gif"	b22
	collectionmeta	.section:Title "section titles"	b23
	collectionmeta	.document:text "entire books"	b24
	collectionmeta	.section:text "chapters"	b25
<hr/>			
(c)	creator	Email address of the collection's creator	
	maintainer	Email address of the collection's maintainer	
	public	Whether collection is to be made public or not	
	beta	Whether collection is beta version or not	
	indexes	List of indexes to build	
	defaultindex	The default index	

38 BUILDING COLLECTIONS

<code>subcollection</code>	Define a subcollection based on metadata
<code>indexsubcollections</code>	Specify which subcollections to index
<code>defaultsubcollection</code>	The default indexsubcollection
<code>languages</code>	List of languages to build indexes in
<code>defaultlanguage</code>	Default index language
<code>collectionmeta</code>	Defines collection-level metadata
<code>plugin</code>	Specifies a plugin to use at build time
<code>format</code>	A format string (explained below)
<code>classify</code>	Specifies a classifier to use at build time

The collection configuration file for the Demo collection is shown in Table 10b. For each collection, the bland configuration file that *mkcol* has produced automatically must be edited into something that looks like this. The most formidable part, the *format* and *collectionmeta* statements, are cosmetic: the basic collection structure is created by the *indexes* statement, which determines what searchable indexes there are, the five *plugin* statements, which determine how the raw input files are read and parsed, and the four *classify* statements, each of which produces one of the four browsing access buttons.

The first block, lines a1–a4 and b1–b4, specifies the creator and maintainer of the collection, whether it is supposed to be made publicly available or not, and whether it is a beta version or not.

The next line specifies the full-text searching indexes that are to be built. Line a5 specifies that there is to be just one index. Indexes can be constructed at the *document*, *section*, and *paragraph* levels. They can contain material from *text* or any metadata—most commonly *Title*. These two elements are separated by a colon in the *indexes* line. Line b5 calls for three indexes to be created: the first for sections, the second for section titles, and the third for complete documents. In addition, text and multiple metadata types can be included in a single index using commas as separators, e.g. *section:text,Title,Creator*.

Collection-level metadata such as the long form of the collection’s name, its icon, the names of indexes, and the “about” text can be defined using *collectionmeta* entries in the configuration file. Lines a12–a13 specify very rudimentary metadata; this has been expanded to lines b20–b25 by adding entries that are largely self-explanatory.

The plugins (lines b7–b11), classifiers (lines b12–b15), and format strings (lines b16–b19) are described in separate sections below. Plugins and classifiers make use of several auxiliary files that will be described in due course: they are collected together in Table 11 for easy reference. Briefly, the *index.txt* file is used by the *IndexPlug* plugin to assign metadata to the

documents; *mags.txt* is a file that is special to this collection that determines which of the books in it are actually magazines, for special treatment in the *titles* browsing list; *sub.txt* specifies the subject hierarchy that appears when you press the *subject* browsing button; and *org.txt* defines the organization hierarchy that appears when you press the *organization* browsing button. These are subsets of the corresponding files for the Humanities Development Library, which accounts for the fact that the numberings do not start from 1. You can see *index.txt* in the *import* directory in Table 5b; the other three files reside in the *etc* directory. (For maximum software reuse, the same format used for configuration files is used for the files that are read in by plugins and classifiers.)

Finally we will describe the subcollection and language-specific index facilities; these are not used in the Demo collection.

Table 11 Auxiliary files used in building the Demo collection

(a) <i>index.txt</i> (used by <i>IndexPlug</i>)	key:	Subject	Organization	Howto	Magazine
	bostid/b22bue	16.11	bostid	"start a butterfly farm"	
	faobetf/fb33fe	14.12	faobfs	<Subject>16.11	
	faobetf/fb34fe	14.12	faobfs	"farm snails"	
	<Subject>16.11				
	bostid/b18ase	16.11	bostid	"introduce little-known Asian ..."	
	bostid/b20cre	16.11	bostid		
	bostid/b17mie	16.11	bostid	"introduce small animals and ..."	
	bostid/b21wae	16.5	bostid	"utilize the ..."	
	<Subject>16.11				
	ecourier/ec158e	23.15	ecc	<Subject>8.1	"<Magazine>The Courier"
	ecourier/ec159e	23.15	ecc	<Subject>6.1	"<Magazine>The Courier"
	ecourier/ec160e	23.15	ecc	<Subject>21.1	"<Magazine>The Courier"
	wb/wb34te	6.4	wb		"achieve gender equality"
(b) <i>mags.txt</i> (used by <i>HDLList</i>)	"The Courier"			3	"The Courier"

40 BUILDING COLLECTIONS

(c)	<i>sub.txt</i>	6	6	"Society, Culture, Community, Woman, Youth, ..."
	(used by <i>Hierarchy</i>)	6.1	6.1	"Social sciences, sociology (works comprising ..."
		6.4	6.4	"Women, gender and development, women's ..."
		8	8	"Communication, Information and Documentation"
		8.1	8.1	"Communication, telecommunication, mass ..."
		14	14	"Agriculture and Food Processing"
		14.12	14.12	"Better Farming series of FAO and INADES"
		16	16	"Animal Husbandry and Animal Product Processing"
		16.5	16.5	"Cattle"
		16.11	16.11	"Other animals (micro-livestock, little known ...)"
		21	21	"Settlements, Housing, Building - ..."
		21.1	21.1	"Settlements and housing: general works incl. ..."
		23	23	"Development Periodicals and Magazines"
		23.15	23.15	"The Courier ACP 1990 - 1996 Africa-Caribbean-..."
<hr/>				
(d)	<i>org.txt</i>	bostid	4	BOSTID
	(used by <i>Hierarchy</i>)	lecho	10	E.C.H.O
		ecc	11	"EC Courier"
		faobfs	15	"FAO Better Farming series"
<hr/>				

Plugins

Plugins are used by the collection-building software to accomplish all the format-specific parsing of each document. The configuration file gives a list of all the plugins to be used in building that collection. For example, the Demo collection (lines b7–b11) specifies five plugins: *GMLPlug*, *HBPlug*, *ArcPlug*, *IndexPlug* and *RecPlug*. During the import operation, each file or directory is passed to each plugin in turn until one is found that can process it—thus earlier plugins take priority over later ones. If no plugin is found that can process the file, a warning is printed (to the error channel) and processing passes to the next file. The same procedure is used during building, this time with the *archives* directory.

Recursion is necessary to traverse directory hierarchies. Although the importing (and building) programs do not perform recursion explicitly, some plugins cause indirect recursion. A plugin may itself pass files or directory names into the plugin pipeline, allowing a directory hierarchy to be traversed. The standard way of doing this is with *RecPlug*, which recurses through a directory structure. If this is present it should be the last element in the pipeline.

General plugins are kept in *perllib/plugins*, while collection-specific ones

are in *collect/collection-name/perl/lib/plugins*. In case of a name conflict, collection-specific plugins override general ones. The plugins that are currently available are listed in Table 12. Only the first three are recursive.

Table 12 Currently-implemented plugins

	Plugin	Purpose
<i>General</i>	ArcPlug	Processes files named in index file <i>archives.inf</i>
	RecPlug	Recurse through a directory structure
	IndexPlug	Assigns metadata from <i>index.txt</i> file
	GMLPlug	Processes GML files generated by <i>import.pl</i>
	TEXTPlug	Processes plain text.
	HTMLPlug	Processes HTML, taking care to replace hyperlinks
	EMAILPlug	Processes email messages, recognizing author, subject, date, etc.
<i>Specific</i>	HBPlug	Processes HTML marked up for UN collections
	FOXPlug	Processes Foxbase dbt files
	PrePlug	Processes HTML output by PRESCRIPT, splitting documents into pages
	GBPlug	Processes Project Gutenberg etext
	TCCPlug	Processes email documents from Computists' Communique

We implied earlier that communication between the *import* and *building* processes is achieved simply by the former placing files in the *archives* directory, and the latter picking them up. In fact, to make the operation more efficient, and to allow the files to be sorted in different ways before building commences, *import* writes a file called *archives.inf* into the top-level directory of the *archives* file structure containing the names of the GML files it has imported. (If *archives.inf* already exists, *import* simply adds to it.) This also allows non-GML files (such as image files) to be ignored, rather than being processed by the plugin pipeline and causing an error message. The function of *ArcPlug* is to check whether it is being passed an *archives.inf*, and, if so, to read the list of files it contains and pass each one in turn into the plugin pipeline. *ArcPlug* should always be included unless *import.pl* will never be used.

As mentioned above, *RecPlug* checks to see if the filename it receives is a directory. If so, each file in that directory is passed into the plugin pipeline. If any of these is itself a directory, it will be picked up by *RecPlug* in the next pass. *RecPlug* is used merely to traverse the directory structure when importing or building directly from original material.

IndexPlug is used to assign metadata to documents from a separate

42 BUILDING COLLECTIONS

metadata file called *index.txt*, which is generally created manually. This plugin checks the filename it receives to see if it is *index.txt*, and if so, reads the list of files, assigns metadata to them as specified, and passes each one in turn into the plugin pipeline. The *index.txt* file associated with the Humanities Development Library (from which the Demo collection is excerpted) contains extra metadata to be associated with each book (Subject classification, Organization, “how to” classification, and Magazine title), and was generated manually from Excel files supplied with the collection. Creating this file was the most time-consuming part of making the collection.

The content of this metadata file is shown in Table 11a. The first line is a key to what metadata fields are included, in this case *Subject*, *Organization*, *Howto*, and *Magazine* metadata. Subsequent lines give the filename of a document followed by its metadata values. For example, the second line of Table 11a assigns to the document that originated in file *bostid/b22bue* the Subject *16.11*, the Organization *bostid*, the “how to” *start a butterfly farm*, and nothing for *Magazine*. A mechanism is provided to allow for multiple field values and missing fields. The next line not only assigns *14.12* to Subject and *faobfs* to Organization by placing field names in angle brackets, but also assigns *16.11* to Subject as well; it makes no assignment to “how to”. All metadata specifications may be repeated in a document file. While this may seem clunky, it combines a mechanism to utilize metadata provided in spreadsheet files with a way of allowing more flexible non-column-oriented specifications.

IndexPlug resembles *ArcPlug* in that only those files that appear in the list will be included. It cannot be used to add metadata to just a few files, for the other files will not be included in the collection. If importing and building take place in separate stages, *IndexPlug* will operate during the importing stage, and the files it produces will be picked up by *ArcPlug* during building.

GMLPlug processes documents in GML format. It should always be included unless *import.pl* will never be used.

TEXTPlug is a plugin for processing plain text. It does not alter the input but places it between `<pre>` and `</pre>` tags for display as preformatted text in an HTML page. It is being extended to allow the input to be GB-encoded, to allow Chinese files to be read.

HTMLPlug processes plain HTML. It replaces hyperlinks: if the target is a document within the collection it makes the appropriate linkage; if not it calls a CGI script that warns the user that they are leaving the collection. (Also being extended with GB-encoding).

EMAILPlug processes email messages in the standard format. It extracts metadata such as *to*, *from*, *subject*, and *date*. It preserves embedded hyperlinks, and turns email addresses into hyperlinks.

HBPlug processes marked-up HTML files supplied for United Nations collections. *HBPlug* is specially written for these collections. It reads the input file, splits it into sections at <<TOC>> tags and replaces <<I>> tags with corresponding HTML tags.

FOXPlug processes a Foxbase DBT file. It provides the basic functionality to read DBT and DBF files and process each record. It should be overridden for a particular database to process the appropriate fields in the file.

The suite of collection-specific plugins is growing rapidly. *PrePlug*, used by the Computer Science Technical Reports collection, processes the simplified HTML output by PRESCRIPT. It splits documents into pages on <!--End Of Page--> or <!--Page No--> tags if present, otherwise at paragraph breaks every 50 lines or so. It also checks whether the document has a corresponding *.info* file, from which it obtains metadata. *GBPlug*, used by the Gutenberg collection, separates out introductory boilerplate and splits documents into pages of 80 or so lines. It also extracts title and author metadata from the text. *TCCPlug* processes the email documents received from the Computists' Communique. It splits documents into separate articles and extracts title and date metadata.

Classifiers

Classifiers are used to create a collection's browsing indexes, for example, the Demo collection's Subject, and Titles A-Z, organization, and "how to" indexes. This information is stored in the GDBM database, and the classifiers are called during the final phase of *buildcol.pl* when this database is being built.

Like plugins, classifiers are specified in a collection's configuration file, one line for each. This line contains the keyword "classify" followed by the name of the classifier and any options it takes. For example, line a11 of Table 10a makes an alphabetic list of the contents of the *Title* metadata field: it takes all documents with this metadata, sorts them by title, and splits them into alphabetical subsections.

A similar command appears as line b13 of Table 10b, and you can see the result by clicking the *Titles A-Z* button of the Demo collection. The *HDList* that it specifies is a classifier written especially for the Humanity Development Library, from which the Demo collection was derived. It is

44 BUILDING COLLECTIONS

identical to the standard *AZList* classifier described below, except that it adds a hierarchy classification for the “Magazines” section, and obtains this information by reading the file *mags.txt* shown in Table 11b. In this case only one document (*The Courier*) is declared as a magazine; in the full Humanity Development Library there are several such documents. This classifier has no options: since it is only used by this collection, the functionality is hard-coded.

The other *classify* lines in the demonstration collection configuration file specify a hierarchical browsing index for the *Subject* metadata (line b12), a hierarchical browsing index for the *Organization* metadata (b14), and a plain list for the “how to” metadata (b15).

Table 13 Currently-implemented classifiers

Classifier	Argument	Meaning
Hierarchy	<i>hfile</i>	Hierarchical classification A classification file
	<i>metadata</i>	Metadata element to test against <i>hfile</i> identifier
	<i>sort</i>	Metadata element to sort documents by (optional)
	<i>buttonname</i>	Name of the button used to access this classifier (defaults to value of <i>metadata</i> argument)
List		A list of documents
	<i>metadata</i>	Include documents containing this metadata element
	<i>buttonname</i>	Name of the button used to access this classifier (defaults to value of <i>metadata</i> argument)
SectionList		A list of sections in documents
AZList		A list of documents split into alphabetical ranges
	<i>metadata</i>	Include all documents containing this metadata element
	<i>buttonname</i>	Name of the button used to access this classifier (defaults to value of <i>metadata</i> argument)
AZSectionList		Like AZList but includes every section of the document
DateList		Like AZList but sorted by date

Classifiers generate a hierarchical structure that is used to display a browsing index. The leaves of the hierarchy are usually documents, but in some classifiers the leaves are sections. The internal nodes of the hierarchy are either *Vlists*, *Hlists*, or *Datelist*s. *Vlists* are lists of items displayed vertically down the page, like the “how to” index in the Demo collection. *Hlists* are displayed horizontally. For example, the *Titles A-Z* display is a two-level hierarchy of internal nodes consisting of an *Hlist* (giving the A-Z selector) whose children are *Vlists*—and their children, in turn, are documents. A *Datelist* is a special kind of *Vlist* that allows selection by month. This is used, for example, in the *Dates* classification of the *Computists’ Weekly* collection.

Classifiers have a metadata argument that determines what documents are included in the hierarchy. Only documents for which this type of data is specified are included. Table 13 lists the classifiers that are available, and their arguments.

All classifiers generate a hierarchy. Some have an arbitrary number of levels that depend on the application data while others have a fixed number of levels. The former are called *Hierarchy* classifiers. For example, the Demo collection's *subject* and *organization* browsers use a multi-level structure of *Vlists* nested within *Vlists* that reflects the particular hierarchy desired; this is controlled by the *hfile* argument (see below). The leaves are documents. There is a further optional argument, *sort*, which determines how the documents at the leaves are ordered. Any metadata can be specified as the sort key. If none is specified, the list will be produced in build order. Ordering at internal nodes is determined by the order in which things are specified in the *hfile* argument.

The second classifier type, *List*, generates a one-level hierarchy consisting of a single *Vlist* whose children are documents—like the “how to” index in the Demo collection. If the *metadata* argument is specified, only documents containing this metadata element are included, and the list is sorted by this metadata. If it is omitted, all documents are included, in build order.

There are, in addition, various varieties of *List* classifiers as shown in Table 13.

- *SectionList*—like *List* but the leaves are sections rather than documents. All sections of the documents are included, except the top level. This is used to create lists of sections (articles, chapters or whatever) as in the *Computists' Communique* collection, where each issue is a single document and comprises several independent news items, each in its own section.
- *AZList*—generates a two-level hierarchy consisting of an *Hlist* whose children are *Vlists*, and their children are documents; the *Hlist* is an A-Z selection list that divides the documents into alphabetic ranges by the metadata type that is specified as an argument. Documents are sorted alphabetically by metadata and the resulting list split into alphabetical ranges.
- *AZSectionList*—like *AZList* but the leaves are sections rather than documents.
- *DateList*—like *AZList*, except that the top-level *Hlist* allows selection by year and its children are *DateLists* rather than *Vlists* (for this classifier the metadata argument defaults to *Date*).

46 BUILDING COLLECTIONS

A final argument for all classifiers is *Buttonname*. The root of the hierarchy corresponds to the button that gives access to the classifier on the Greenstone navigation bar. The name of this button can be set by *Buttonname*; it defaults to the metadata argument. Buttons are provided for each Dublin Core metadata type, and for some other types of metadata. (Soon they will be generated on demand by Gimp, a scriptable image-manipulation program.)

Each classifier is given an implicit name generated from its position in the configuration file. For example, the third classifier specified is called CL3. This is used to name the GDBM fields that contains data that defines the classifier hierarchy.

In addition, the query results list that is returned by a search is also structured as a one-level hierarchy, comprising a single *Vlist*.

Finally, before we leave the topic of Classifiers we describe the format of the file specified by *hfile* in *Hierarchy* classifiers, which gives a textual description of the hierarchy. In the case of the Demo collection, line b12 of Table 10b specifies the file *sub.txt*, which is shown in Table 11c (and was generated from an Excel file supplied with the Humanities Development Library collection). There is one line for each section, comprising three fields:

- identifier, which matches the specified metadata value for each section to be included;
- position-in-hierarchy identifier, in multi-part numeric form—e.g. 1, 1.2, 2.6.1;
- title of section.

This example is a slightly confusing one because the number representing the hierarchy appears twice on each line. The metadata type “Hierarchy” stores its values in hierarchical numeric form, which accounts for the first occurrence. It is the second occurrence that is used to determine the hierarchy that the browser implements.

A second example is the organization hierarchy of the Demo collection, shown in Table 11d as file *org.txt*—which is smaller and simpler than *sub.txt* because this metadata has a simpler structure than the subject hierarchy. Here the metadata type *Organization* contains acronyms such as *accu*. The hierarchy has just one level, so the position-in-hierarchy identifier is just an integer. The title does not need to be quoted if it does not contain spaces.

Format strings

Format strings govern how the items produced by the classifiers are to be displayed on the screen. They are introduced by the keyword *format*, followed by an item to which this specification is to apply, followed by an actual format specification itself. There are two ways of declaring which items a format string is to apply to. The first specify how particular kinds of items will appear—such as the document text, and other aspects of the interface. The second way determines how the *Hlists*, *Vlists*, and *DateLists* that appear in classifier hierarchies are to be displayed.

Table 14 shows the possible options for the first kind of format string. The *DocumentButtons* option controls what buttons are to be displayed on a document page. Here, *string* is a list of buttons to be displayed (separated by |), possible values being *Detach*, *Highlight*, *Expand Text*, and *Expand Contents*. Reordering the list reorders the buttons.

Table 14 The format options

format DocumentIcon	string/true/false	If true, put cover image at top left of document page; if a format string, this is the heading to go at the top left
format DocumentContents	true/false	Display table of contents (if document is hierarchical), or next/previous section arrows and “page k of n” text (if not).
format DocumentButtons	string	Controls the buttons that are displayed on a document page (default <i>Detach/Highlight/Expand Text/Expand Contents</i>)
format DocumentText	formatstring	Format of the text to be displayed on a document page (default <i>[Text]</i> , see below)
format DocumentArrowsBottom	true/false	Display next/previous section arrows at bottom of document page (default <i>true</i>).

The second kind of format string gives a flexible way of controlling how different classifiers are displayed. A single specification can apply to all nodes of a given type in any classifier, or to all nodes in a particular classifier, or to a particular kind of node in a particular classifier.

Format specifications are introduced by the word *format* followed by a two-part keyword. Both parts are optional. The first part, if present, identifies the classifier by its positional code, CL1, CL2, CL3, ... for the first, second, third, ... classifier specified in the collection configuration file. As noted earlier, the query results list that is returned by a search is also structured as a one-level hierarchy, comprising a single *Vlist*; this can be specified by using the word *Search* as the first part of a format keyword. The second part, if present, identifies whether the format specification is to apply to *Hlists*, *Vlists*, or *DateLists*. For example:

48 BUILDING COLLECTIONS

format CL4Vlist ...	applies to all <i>Vlists</i> in CL4
format CL2Hlist ...	applies to all <i>Hlists</i> in CL2
format CL1DateList ...	applies to all <i>DateLists</i> in CL1
format SearchVlist ...	applies to the Search Results list
format CL3 ...	applies to all nodes in CL3, unless otherwise specified
format Vlist ...	applies to all <i>Vlists</i> in all classifiers, unless otherwise specified

The “...” in the above format strings are HTML format specifications that control the information, and its layout, that appear on the Web pages that display the classifier. As well as HTML specifications, any metadata may appear within square brackets: its value is interpolated in the appropriate place. Lastly, any of the items shown in Table 15 may appear in format strings. The syntax for the strings also includes a conditional statement, which is illustrated in an example below.

Table 15 Items that may appear in format strings

[Text]	The document’s text
[link]	The HTML to link to the document itself
... [/link]	
[icon]	An appropriate icon (usually the little text icon when in results string)
[num]	The document result number (useful for debugging).

Recall that all classifiers produce hierarchies. Each level of the hierarchy is displayed in one of five possible ways: as a *Hlists*, *Vlists*, or *DateLists*; as a *PagedList*; or as *Invisible*. The very top levels of hierarchies are always displayed as *Invisible*, because the name of the classifier is already shown separately on the Greenstone navigation bar using the button specified by *Buttonname*.

Examples of classifiers and format strings

Consider first the Demo collection’s “how to” classification. This is the fourth classifier specified in the collection configuration file (line b15 in Table 10b), and is therefore referred to as CL4. The classifier specification and corresponding format statement (line b17 in Table 10b) are reproduced below. The “how to” information is generated from the *List* classifier, and its structure is a simple list of titles.

```
classify          List metadata-Howto
format CL4Vlist  "<br>[link][Howto][ /link]"
```

The top level of the hierarchy is “displayed” as *Invisible* (always used for

the top level). Its children are displayed as a *Vlist* (vertical list), which simply lists the sections vertically. Each element of the list is on a newline (“
”) and contains the *Howto* text, as a link to the document itself.

The *Subject* classification in the Demo collection is the first one specified (line b12 in Table 11b) and therefore denoted CL1; the *Organization* classification is the third one (line b14 in Table 10b), CL3. Both specifications are reproduced below; they are generated by the *Hierarchy* classifier and therefore comprise a hierarchical structure of *Vlists*.

```
classify Hierarchy hfile=sub.txt metadata=Subject
sort=Title

classify Hierarchy hfile=org.txt metadata=Organization
sort=Title
```

The final classification for the Demo collection is *Titles A-Z* (CL2). This is generated by a special classifier, *HDLList*, that is the same as an *AZList* but has a “magazines” section added to it:

```
classify HDLList Title
```

This accounts for the four *classify* lines in Table 10b.

Coincidentally, there are also four *format* lines. We have already discussed one, the *CLAVlist* specification. The remaining three are the first type of format string documented in Table 14. For example, line b19 of Table 10b formats the actual document text, with the title of the relevant chapter or section preceding the text itself. Line b18 puts the cover image at the top left of each document page.

Line b16 of Table 10b contains a rather complicated specification that formats the query result list returned by a search. A simplified version of the format string is

```
"<td valign=top>[link][icon][link]</td>
 [link][Title][link]</td>"
```

This is designed to appear as a table row, which is how the query results list is formatted. It gives a small icon linked to the text, as usual, and the document title, hyperlinked to the document itself.

In this collection, documents are hierarchical. In fact the above hyperlink anchor will be the title of the section that the query returns. However, it would be better to augment it with the title of the enclosing section, the enclosing chapter, and the book in which it occurs. There exists a special metadata item, *parent*, which is not stored in documents but is implicit in

50 BUILDING COLLECTIONS

any hierarchical document, that can produce such a list. This will either return the parent document, or if used with the qualifier *All*, the list of hierarchically enclosing parents, separated by a character string that can be given after the *All* qualifier. Thus

```
"<td valign=top>[link][icon][link]</td>
<td>{[parent(All': `):Title]: }[link][Title][link]</td>"
```

will have the desired effect of producing a list of the book title, chapter title, etc that enclose the target section, separated by colons, with a further colon followed by a hyperlink to the target section's title.

Unfortunately, if the target is itself a book, there is no parent and so an empty string will appear followed by a colon. To remove this requires a conditional statement. For this reason, conditional *if* and *or ... else* statements can be placed in format strings:

```
{If}{[metadata], action-if-non-null, action-if-null}
{Or}{action, else another-action, else another-action, etc}
```

In both cases, curly brackets are used to signal that the statements should be interpreted and not just printed out as text. The *If* tests whether the metadata is empty and takes the first clause if not, otherwise the second one (if it exists). Any metadata item can be used, including the special metadata *parent*. The *Or* statement evaluates each action in turn until one is found that is non-null. That one is sent to the output and the remaining actions are skipped. Currently, *If* and *Or* statements cannot be nested.

Returning to line b16 of Table 10b, the full format string is

```
"<td valign=top>[link][icon][link]</td>
<td>{If}{[parent(All': `):Title],
        [parent(All': `):Title]:
} [link][Title][link]</td>"
```

This precedes the *parent* specification with a conditional that checks whether the result will be empty, and only outputs the parent string if it is present. Incidentally, the *parent* can be qualified by *Top* instead of *All*, which gives the top-level document name that encloses a section—in this case, the book name. No separating string is necessary with *Top*.

Some final examples will illustrate other features. A *DateList* is used in the *Dates* classification of the Computists' Weekly collection (which happens to be the second classifier, CL2). The *DateList* classifier differs from *AZList* in that it always sorts by Date metadata, and the bottom branches of the browsing hierarchy use *DateList* instead of *Vlist*, which causes the year and month to be added at the left of the document listings.

```
classify AZSectionList metadata=Creator
```


52 BUILDING COLLECTIONS

values are specified using the ISO 639 standard two-letter codes for representing the names of languages—for example, *en* is English, *zh* is Chinese, and *mi* is Maori. Since metadata values can be specified at the section level, parts of a document can be in different languages.

For example, if the configuration file contained

```
indexes section:text section:Title document:text
paragraph:text
languages en zh mi
```

section text, section title, document text, and paragraph text indexes would be created for English, Chinese, and Maori—twelve indexes altogether. Adding a couple of subcollections multiplies the number of indexes again. Care is necessary to guard against index bloat.

(This index specification could be defined using the *subcollection* facility rather than the *languages* facility. However, since the current syntax precludes creating subcollections of subcollections, it would then be impossible to index each language in the subcollections separately.)



D

Installing the Greenstone software

We now turn to some more technical aspects of the software. To make your computer into a Greenstone server, you need to install the Greenstone software, and we first explain how to do this under Unix and Windows operating systems. Under Unix, it is easy to install the full software; under Windows you can install different parts of it depending on what you want to do. We assume that you already have Web server software installed. Greenstone uses three packages that you will have to obtain from elsewhere:

- STL, the C++ standard template library (www.sgi.com/Technology/STL)
- PERL 5, the PERL programming language (www.gnu.org/software/perl)
- GDBM, the GNU database system (www.gnu.org/software/gdbm) (the Windows version is included with Greenstone in `packages/wingdbm`)

MG, the Managing Gigabytes full-text indexing program, is contained within the Greenstone software and will be installed as part of the installation procedure. Greenstone can use the Fast CGI system to speed up operation. If this is not installed, it will work as a regular CGI script. Any bugs or installation problems should be reported to greenstone@cs.waikato.ac.nz.

Having explained how to install the software, we describe in Section D.3 how to make a standalone CD-ROM containing particular collections that can then be distributed for others to use. Then we examine the configuration files that allow you to configure Greenstone to run under your system, and follow this up by a walk through the directory structure containing the Greenstone software. The next section, D.6, describes the user logging facility, and following that we look at the maintenance and

54 INSTALLING THE GREENSTONE SOFTWARE

administration facilities that are built into the Greenstone system. It is easy to translate the interface into different languages, and section D.7 tells you what you need to know in order to extend Greenstone to accommodate languages that are not yet supported.

D.1 Installing on Unix

Here is the procedure for installing the Greenstone software on Unix.

1. Download the `gsdl` distribution (currently version 2.11, in file `gsdl-2.11.tar.gz`) from www.nzdl.org/technology
2. Extract the gzipped `tar` archive and `cd` to the `gsdl` directory it creates

```
type tar xvzf gsdl-2.11.tar.gz
cd gsdl
```
3. Run the configure script using `./configure` to check the resources on your system and configure the Greenstone software accordingly.

```
type ./configure
```
4. Compile the Greenstone software by running `make`.

```
type make
```
5. Install the software by running `make install`. This installs the newly-compiled binaries to all the right places; it places the compiled executable file in the `$FSDLHOME/cgi-bin` directory.

```
type make install
```
6. To build the demonstration collection that comes with the distribution, run the `bulddemo.sh` script from within the `gsdl` directory.

```
type bulddemo.sh
```
7. The `cgi-bin/gsdlsite.cfg` must be altered to suit your site. Typically you'll need to edit the `httpprefix` (the Web path to the `gsdl` directory), `httpimg` (the path to `gsdl/images`), and `gsdlhome` fields.

```
edit cgi-bin/gsdlsite.cfg appropriately
```
8. Move all the files in `$GSDLHOME/cgi-bin` directory to your system's `cgi-bin` directory, to make them active.

```
type mv cgi-bin/* /usr/local/apache/cgi-bin
```
9. Using a Web browser, look at the URL `.../cgi-bin/library`. This is the home page of your Greenstone system. The small test collection `demo` will be the only collection shown at this stage.
10. Download other collections from www.nzdl.org. The collections available are shown in Table 16, along with approximate sizes (tar'd and gzipped).

Table 16 Collections and their sizes

collection	abbrev	built size (Mb)	download size (Mb)
Computer science bibliography	csbib	866	?
Computer science technical reports	cstr	2010	~1800
Project Gutenberg	gutenberg	432	457
HCI bibliography	hcibib	36	5
Humanity Development Library	hdl	199	387
Indigenous Peoples	ipc	7.5	4
Maori newspapers	niupepa	12	?
Oral history	ohist	2.5	?
The computists weekly	tcc	21	8
Tidbits magazine	tidbits	10	5
United Nations University collection	unu	52	?
Women's history	whist	6	?
		3654	

D.2 Installing on Windows

The Greenstone software works equally well under Windows and Unix.

Installing binaries

There are three different types of library executable for windows:

1. A single-user version for collections on CD-ROM or hard drive. This cannot be accessed from other machines.
2. A server version of the above which can be accessed from other machines. This contains its own self-contained Web server.
3. A CGI application for running on an existing Web server (Apache, IIS or similar).

The source code for these versions is very similar. The only difference between the first two is that one line of code is commented out. The difference with the third is that the makefile is edited to replace a couple of *.cpp* and *.h* files.

The third version provides exactly the same facilities as the Unix system and can be used to serve collections in just the same way. Two files are to be downloaded and placed in the system's *cgi-bin* directory: *library.exe* and *gsdlsite.cfg*. Edit the site configuration file to set the *\$GSDLHOME* variable to the GSDL home directory on your system.

Installing the source code

If you want to create new collections or work on the Greenstone software itself, you will need to install the source code. This gives you the ability to

- build collections
- modify the building or library (interface) code to suit your needs
- compile the library to run through a web server
- compile the library to run as a stand-alone program off cd-rom or hard disk.

Windows installation is very similar to the Unix procedure. However, we do not yet have an equivalent to the configure script described above for Unix—we have been concentrating instead on making the Unix installation as smooth as possible. Eventually we will provide an easily-installable package using InstallShield. However, at the moment it must be done manually.

STL, the C++ Standard Template Library, will need to be downloaded and installed; there are several Windows ports available. In fact, despite its name, there are different versions of the Standard Template Library. We recommend *STLport-3.2*, which you can download from www.stlport.org. (We have had some trouble with other versions, such as Objectspace STL.) To build collections, the PERL programming language must be downloaded and installed.

Having installed STL, download the appropriate *gsdl* package from www.nzdl.org/technology and unzip it. To compile the library, perform the following steps using *nmake* (or *make* on some systems) in the *\$GSDLHOME* directory. Compilation is known to work with the Microsoft C++ compiler; other compilers may require some modifications to be made to the software.

1. Compile *wingdbm*. This is a port of the GDBM database software and resides in *gsdl/packages/gdbm*.

```
type cd gsdl\packages\gdbm
      nmake win32.mak
```

2. Compile the *gsdl/lib* directory. This code is shared by both the library and some of the building utilities.

```
type cd gsdl\lib
      nmake win32.mak
```

3. Compile and install *txt2db.exe*. This is the utility used to create GDBM databases while building a collection, and resides in *gsdl/src/txt2db*. It will be installed into *gsdl/bin/windows*.

```
type cd gsdl\src\txt2db
```


D.3 MAKING A GREENSTONE CD-ROM 57

- ```
 nmake win32.mak
 nmake install /F win32.mak
```
4. Compile and install *db2txt.exe*, a utility for viewing a GDBM database as text which is useful when debugging a new collection.
- ```
                                type cd gsd\src\db2txt
                                nmake win32.mak
                                nmake install /F win32.mak
```
5. Compile and install *hashfile.exe*, a utility used for generating unique identifiers for each document while importing and building.
- ```
 type cd gsd\src\hashfile
 nmake win32.mak
 nmake install /F win32.mak
```
6. Compile and install *mg*, in *gsd/packages/mg*. Compilation creates several *.exe* files in *gsd/packages/mg/src/text*; they should be moved manually to the *bin* directory.
- ```
                                type cd gsd/packages/mg/lib
                                nmake win32.mak
                                copy gsd/packages/mg/src/text/*.exe gsd/bin/windows
```
7. The *cgi-bin/gsdlsite.cfg* must be altered to suit your site. Typically you'll need to edit the *httpprefix* (the Web path to the *gsd* directory), *httpimg* (the path to *gsd/images*), and *gsdlhome* fields.
- ```
 type move etc\gsdlsite.cfg C:\apache\cgi-bin
 edit cgi-bin/gsdlsite.cfg appropriately
```
8. Compile *library.exe* for Windows, in the form of a CGI script for a Web server—as when running under Unix. (An alternative is to compile with web serving software included, as when putting software onto CD-ROM; see next section). Make the collection server and then the receptionist; this creates a *library.exe* file in *gsd/src/recpt* that you should move (temporarily) to *gsd/cgi-bin*.
- ```
                                type cd gsd/src\colservr
                                nmake win32.mak
                                cd gsd/src/recpt
                                nmake win32.mak
                                move gsd/src/recpt\library.exe gsd/cgi-bin
```
9. Now move all the files in *\$GSDLHOME/cgi-bin* directory into to your system's *cgi-bin* directory, to make them active.
- ```
 type move gsd/cgi-bin/* C:\apache\cgi-bin
```
- (Alternatively you can just execute *gsd/cgi-bin/library.exe* directly from the command line.)

## D.3 Making a Greenstone CD-ROM

The process of putting a Greenstone collection on to a CD-ROM involves three stages. First, it is necessary to install the Greenstone software as described above, except that, in step 8, *library.exe* is compiled with web serving software included. Second, the collection is built in the ordinary way. Third, a CD-ROM installation package is created; we use the standard InstallShield package for this.

## 58 INSTALLING THE GREENSTONE SOFTWARE

The first stage involves replacing step 8 above. Before doing this, you need to create a new project using Visual C++, a fairly lengthy process. Bring up the *Project Settings* dialog box:

Under C/C++, ensure that the following preprocessor definitions are in place:

```
WIN32, SHORT_SUFFIX, _LITTLE_ENDIAN, __WIN32__, PARADOCNUM,
HAVE_CONFIG_H, GSDL_NAMESPACE_BROKEN
```

If using STLport (this will differ for other STL packages):

```
__STL_NO_NEW_IOSTREAMS, GSDL_USE_IOS_H
```

Include the following directories in the search path (put */I "directory"* under *project options*):

```
gsdl\packages\mg
gsdl\packages\mg\lib
gsdl\packages\mg\src\text
gsdl\packages\wingdbm
gsdl\lib
gsdl\src\colservr
gsdl\src\recpt
gsdl\src\w32server
the path to STL (e.g. C:\STLport-3.2\stlport)
```

Under *Link*, type in the output filenames, e.g.

```
gsdl\library.exe, gsdl\librarydebug.exe
```

Include the following library modules:

```
gsdl\packages\mg\lib\libmg.lib
gsdl\packages\mg\src\text\libtextin.lib
gsdl\packages\wingdbm\gdbm.lib
any libraries required by STL
```

Next insert all the necessary files into the project, as follows.

From *gsdl/lib*:

```
fileutil.cpp, gsdlunicode.cpp, display.cpp, gsdltimes.cpp,
gsdlconf.h, cfgread.cpp, text_t.cpp.
```

From *gsdl/src/colservr*:

```
colservrconfig.cpp, mgq.c, phrasesearch.cpp, source.cpp,
filter.cpp, querycache.cpp, mgsearch.cpp, browsefilter.cpp,
maptools.cpp, queryfilter.cpp, collectserver.cpp,
mggdbmsource.cpp, queryinfo.cpp.
```

From *gsdl/src/recpt*:

### D.3 MAKING A GREENSTONE CD-ROM 59

```
OIDtools.cpp, converter.cpp, invbrowserclass.cpp,
action.cpp, datelistbrowserclass.cpp, recptproto.cpp,
authenaction.cpp, documentaction.cpp, nullproto.cpp,
statusaction.cpp, browserclass.cpp, extlinkaction.cpp,
pageaction.cpp, tipaction.cpp, browsetools.cpp,
formattools.cpp, pagedbrowserclass.cpp,
hlistbrowserclass.cpp, pingaction.cpp, usersaction.cpp,
cgiargs.cpp, htmlbrowserclass.cpp, queryaction.cpp,
vlistbrowserclass.cpp, cgiutils.cpp, htmlgen.cpp,
querytools.cpp, htmlutils.cpp, receptionist.cpp,
comtypes.cpp, infodbclass.cpp, recptconfig.cpp,
delhistoryaction.cpp, historydb.cpp.
```

From *gsdl\src\w32server*:

```
netio.cpp, server.ico, wincgiutils.cpp, cgiwrapper.cpp,
fnord.cpp, httpsrv.cpp, server.rc, gsdlcol.bmp,
newgsdl.bmp, settings.cpp, conftools.cpp, httpreq.cpp,
locate.cpp, parse.cpp, startbrowser.cpp, d_winsock.cpp,
httpsend.cpp, resource.h.
```

To complete this first stage, replace step 8 of the previous section with this:

8. Use Visual C++ to compile the library to create the output file as specified under *Link* above (e.g. *gsdl\library.exe*).

The second stage, having set up the software as described above, is to build the collection you wish to put on to CD-ROM. This is done just as described in Section C of this manual. For the Demo collection, for example, you need to execute *import.pl* to import it, then *buildcol.pl* to build it, and finally remove the old contents of the *index* directory and move the contents of the *building* directory into it.

```
type import.pl demo
build.pl demo
del gsdl\collect\demo\index\
move gsdl\collect\demo\building* gsdl\collect\demo\index
```

The third and final stage is to create a CD-ROM installation package for the digital library software and the collection itself using InstallShield. This involves the following steps.

1. Edit *gsdl\src\checkis\checkis.cpp* and compile *checkis.exe*, which is the small program run automatically when the CD-ROM is inserted in the drive to check the registry to see if the CD is already installed (if not it runs *setup.exe*, otherwise it asks the user whether to run the library). Lines 16–17 set the names of the registry keys to check the collection and volume names; they will need to be altered appropriately. Line 59 gives the message to display when asking the user whether to run the library; this may also need to be changed.

## 60 INSTALLING THE GREENSTONE SOFTWARE

2. Use InstallShield project (as described below) to create a *disk1* directory.
3. Use CD-writing software to write the contents of *disk1* on to CD-ROM.

Creating an initial InstallShield project is a fairly involved process; however, once done it is easily edited for each new CD-ROM to be created. (We can supply something to get you started). The main tasks are as follows.

- Edit the string resources used by the project. These include strings used by the dialog boxes during the setup process, and collection/volume name registry keys—which should correspond to those used in *checkis.exe*.
- Insert the appropriate files into the project—all files to be placed on the CD-ROM, including *library.exe*, *checkis.exe*, *autorun.inf*, *README.txt*, all collection files, macro files, indexes, etc.
- Compile the InstallShield script and build the media, a lengthy step if many files have been included. Once built, a single directory (normally *disk1*) contains all files to be placed on the CD-ROM (including ones generated by Installshield, like *setup.exe*).

### D.4 Configuration files

The operation of the software is controlled by several configuration files. We have encountered the collection-specific configuration file *collect.cfg* in Part C when learning how to build collections, and the format of this file was described there. The Greenstone system includes other configuration files, all of which are read using the same software and so have the same format.

The receptionist can run in a “general” mode or in a “collection-specific” mode. In the former model it reads two configuration files, in this order:

```
../cgi-bin/gsdlsite.cfg
etc/main.cfg
```

The site configuration file *gsdlsite.cfg*, which resides in the system’s *cgi-bin* directory, is used to configure the Greenstone software for the site where it is installed. Typical entries in these files are shown in Table 17. It is designed for keeping configuration options that are particular to a given site, like *httping*, *gwsgi* and *\$GSDLHOME*.

The main configuration file *main.cfg* contains information that is common to all collections served by the receptionist. It resides in

*\$GSDLHOME/etc*.

In “collection-specific” mode the receptionist will read four configuration files:

```

.../cgi-bin/gsdlsite.cfg
collect/collection-name/etc/site.cfg
etc/main.cfg
collect/collection-name/etc/collect.cfg

```

As their name implies, collection configuration files are collection-specific, and may override settings in the main configuration files.

Table 17 Entries in the system configuration files

|                     |                          |                                                                                                                                       |
|---------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>gsdlsite.cfg</i> | <code>httpprefix</code>  | HTTP address of <i>\$GSDLHOME</i>                                                                                                     |
|                     | <code>httpimg</code>     | HTTP address of the directory containing the images used in the interface                                                             |
|                     | <code>gwcgi</code>       | HTTP address of the CGI script being run. (If the HTTP server sets the environment variable <i>\$SCRIPT_NAME</i> , this overrides it) |
|                     | <code>gsdlhome</code>    | Directory that corresponds to <i>\$GSDLHOME</i>                                                                                       |
| <i>etc/main.cfg</i> | <code>maxrequests</code> | Maximum number of requests a Fast CGI process will serve before it exits                                                              |
|                     | <code>maintainer</code>  | Email address of the receptionist’s maintainer                                                                                        |
|                     | <code>macrofiles</code>  | Display macro files that are to be loaded by the receptionist                                                                         |
|                     | <code>status</code>      | Status page enabled/disabled                                                                                                          |
|                     | <code>logcgiargs</code>  | Enables/disables logging of user activity                                                                                             |
|                     | <code>usecookies</code>  | Determines whether “cookies” are used to identify users                                                                               |

## D.5 Where to find the software

The Greenstone software is stored in the directories shown in Table 18. In the *src* directory, the *receptionist* is the program that organizes the Greenstone user interface and presents it to the user over the Web interface, while one *collection server* process is created for each collection to manage interaction with it. The other programs are used at built time: *txt2db* to create the GDBM database and *hashfile* to generate the OID of a document. Each collection’s GDBM file is stored as a *.bdb* file (or *.ldb* on little-endian machines) in the collection’s *index/text* directory, and the *db2txt* program is a useful tool for viewing a GDBM database as text. The executable versions of these utilities are stored in *bin/\$GSDL*.

Moving to the *bin* directory, *\$GSDL* is a shell variable that identifies the operating system being used: this is necessary when a file system is shared between machines with different operating systems. Possible values are *linux*, *sunos*, and *windows*, and the appropriate value is set in

## 62 INSTALLING THE GREENSTONE SOFTWARE

Table 18 The directories that contain the Greenstone software system

|            | Directory                                                                     | Contents                                                                                                                                                                                                                                                                                                |
|------------|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$GSDLHOME | /src                                                                          | C++ source code                                                                                                                                                                                                                                                                                         |
|            | /recpt                                                                        | Program that organizes the Greenstone user interface (approx 11,500 lines of C++)                                                                                                                                                                                                                       |
|            | /colservr                                                                     | Program responsible for providing access to a collection when it is being used (approx 5,500 lines)                                                                                                                                                                                                     |
|            | /txt2db                                                                       | Program used at build time to create the GDBM database                                                                                                                                                                                                                                                  |
|            | /db2txt                                                                       | Tool for viewing a GDBM database as text                                                                                                                                                                                                                                                                |
|            | /hashfile                                                                     | Program used at import or build time to generate document OIDs                                                                                                                                                                                                                                          |
|            | /bin                                                                          | Executable code                                                                                                                                                                                                                                                                                         |
|            | /\$GSDL                                                                       | Binaries                                                                                                                                                                                                                                                                                                |
|            | /script                                                                       | Perl scripts                                                                                                                                                                                                                                                                                            |
|            | /cgi-bin                                                                      | All CGI scripts, to be moved to the system <i>cgi-bin</i> directory                                                                                                                                                                                                                                     |
| /perllib   | /plugins                                                                      | Perl modules used at build time<br>The plugins for handling documents of different formats. The build software will look here first when searching for plugins to build this collection. This allows for collection specific plugins to be written, existing plugins to be overridden and changed, etc. |
|            | /classify                                                                     | The classifiers for sorting documents into different classification types (e.g. a list sorted by title, a hierarchical classification etc).<br>Collection specific classifiers (same as for plugins).                                                                                                   |
| /lib       | C++ code used by both receptionist and collection server (approx 4,000 lines) |                                                                                                                                                                                                                                                                                                         |
| /packages  | /fcgi                                                                         | Source code for Fast CGI                                                                                                                                                                                                                                                                                |
| /packages  | /mg                                                                           | Source code for MG                                                                                                                                                                                                                                                                                      |
| /unicode   |                                                                               | Unicode translation tables for the GB Chinese character set                                                                                                                                                                                                                                             |
| /etc       |                                                                               | Configuration files, initialization and error logs, authorization databases                                                                                                                                                                                                                             |
| /macros    |                                                                               | Display macro files                                                                                                                                                                                                                                                                                     |
| /images    |                                                                               | Images used by interface                                                                                                                                                                                                                                                                                |
| /docs      |                                                                               | Documentation                                                                                                                                                                                                                                                                                           |

the *setup* script. The Perl scripts are stored in files whose names end in *.pl*. Although this is unnecessary for Unix systems, Windows systems need the extension in order to execute these scripts properly.

At present (because as yet only the null protocol is implemented) the receptionist and collection server code are compiled together to generate a single executable file called *library* for the receptionist and collection servers together. This is installed temporarily in the *cgi-bin* directory, the contents of which must be moved into the system-dependent location for CGI scripts. The *cgi-bin* directory also contains the site configuration file *gsdlsite.cfg*, and any other CGI scripts that are necessary.

Turning to the *perllib* and *lib* directories, the Perl and C++ source code both make substantial use of libraries. Examples of C++ code used by both receptionist and collection server include *text\_t.cpp* and *text\_t.h*, which implement the *text\_t* string class used throughout all the code, and *display.cpp* and *display.h*, which implement the *display* class for manipulating and displaying the macro language that is used to specify the contents of Web pages.

Some packages are also used by the system. Although it is really an external package, the Fast CGI source code needs to be located with the Greenstone software because references to it are compiled in with the code. MG is the *Managing Gigabytes* scheme for full-text indexing and search; the binaries are installed to *bin/\$GSDL*OS.

Other subdirectories contain miscellaneous information used in the Greenstone system. *etc* contains the top-level configuration file *main.cfg*, the authorization databases *key.db* and *users.db* (both in GDBM format), and error logs. *macros* contains macro files that are used for text display. *images* contains all the images used by the interface. *docs* contains documentation, including this manual and *makecol.txt*, a short beginners guide to setting up new collections.

## D.6 User logs

All user activity—every page that each user visits—can be recorded by the Greenstone software. Logging is enabled by including the lines

```
logcgiargs true
usecookies true
```

in the appropriate configuration file. This is normally done in *main.cfg* to turn on logging for all collections. (Currently it is not possible to control logging at the collection level). Both options are false by default, so that

## 64 INSTALLING THE GREENSTONE SOFTWARE

no logging is done unless they are set. It is the *logcgiargs* line that actually turns logging on and off. However, without *usecookies* the user's identity will not be recorded.

The log file is placed in *\$GSDLHOME/etc/usage.txt*. Each line pertains to a page visit. It contains five components: first the path to the receptionist program (because several receptionists may share a log file), then the IP address of the user's computer, then a timestamp in square brackets, then the CGI arguments in parentheses, and finally the name of the user's browser (Netscape is called "Mozilla"). Here is a sample line, split for ease of reading into these components (though it would contain no line breaks in the log file).

```
/fast-cgi-bin/niupepalibrary
its-www1.massey.ac.nz
[950647983]
(a=p, b=0, bcp=, beu=, c=niupepa, cc=, ccp=0, ccs=0, cl=,
cm=, cq2=, d=, e=, er=, f=0, fc=1, gc=0, gg=text, gt=0, h=,
h2=, hl=1, hp=, il=1, j=, j2=, k=1, ky=, l=en, m=50, n=,
n2=, o=20, p=home, pw=, q=, q2=, r=1, s=0, sp=frameset,
t=1, ua=, uan=, ug=, uma=listusers, umc=, umnpwl=, umnpw2=,
umpw=, umug=, umun=, umus=, un=, us=invalid, v=0, w=w, x=0,
z=130.123.128.4-950647871)
"Mozilla/4.08 [en] (Win95; I ;Nav)"
```

The last CGI argument, "z", gives the value of the cookie: it comprises the user's IP number followed by the timestamp when they first accessed the digital library.

## D.7 Maintenance and administration

There is a maintenance and administration facility associated with each Greenstone installation. This provides a variety of services, including viewing on-line logs of internal errors encountered by the system, maintaining and updating collections, accessing technical information such as CGI arguments, and adding new users. This facility can be accessed from *www.nzdl.org*, but it is "invisible"—there is no visible link. To enter it, click a hidden button below the link to PDF information sheets under the "New Zealand Digital library Project" heading. The following sections correspond to the menu items displayed at the lefthand side of every Greenstone *Maintenance and administration* page (for example, Figure 2).

### User management

Greenstone has the facility for collections to perform authentication actions before returning information to users, by requesting a user name



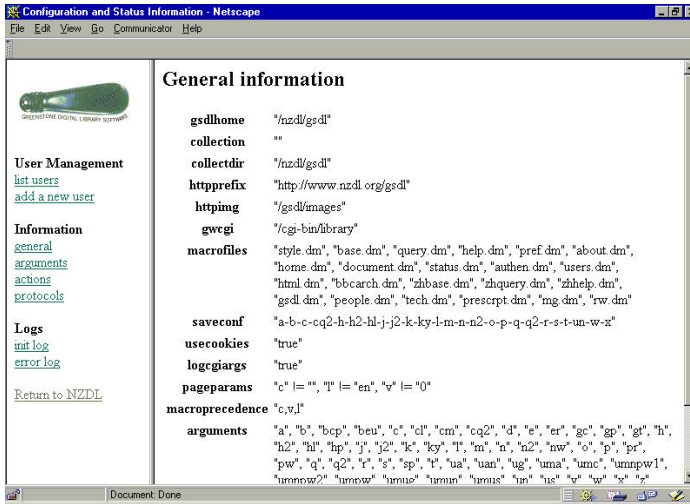


Figure 2 The general information page

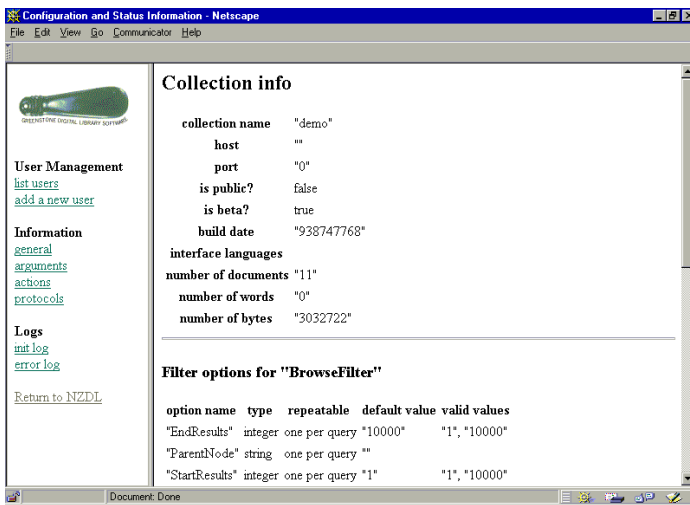


Figure 3 Information displayed for the Demo collection

and password. Current collections do not use this facility. However, users can be listed and altered from the administration page. The ability to do this is of course also protected: only users who have administrative privileges can add new users. It is also possible for each user to belong to a list of "groups". At present, the only extant group is "administrator", membership of which allows one to add and remove users etc.

User information is recorded in two GDBM databases: *etc/users.db* and *etc/key.db*. The first contains all information relating to users. The second contains temporary keys that are created for each page access. These expire after ten minutes (five minutes for the administrators).

Currently, the system prevents access to the status action by anyone not belonging to the "administrator" group. It would be equally simple to prevent access to other parts of the receptionist, and to create public and private versions of collections.

## Information

You can view a general information page like that in Figure 2. This summarizes various information about the Greenstone installation, most of which will be obvious to those who work with the software.

A *protocols* menu item gives information about each of the collections offered by the system; clicking a particular collection name brings up information about that collection, gathered from its collection configuration file. Figure 3 shows the information displayed for the Demo collection.

The user interface code (i.e. the “receptionist”) uses *actions* to communicate the wishes of the user. These actions correspond to the CGI argument labeled *a*. For example, if *a=status* the receptionist invokes the *status* action (which displays the status page). Table 19 shows the principal actions currently supported by Greenstone. A menu item gives access to lists of all actions supported by the system, and another leads to the arguments that these actions take.

Table 19 Principal actions supported by Greenstone

| Action        | Meaning                                                                                                      |
|---------------|--------------------------------------------------------------------------------------------------------------|
| <i>a</i>      | The authentication action                                                                                    |
| <i>d</i>      | The document action (for displaying document text, table of contents etc.)                                   |
| <i>p</i>      | The page action (for displaying general interface pages like the help page, preferences page etc.)           |
| <i>ping</i>   | The ping action (for pinging a collection server to test for response)                                       |
| <i>q</i>      | The query action (for doing all searching)                                                                   |
| <i>status</i> | The status action (for displaying information about the receptionist and any collection servers it knows of) |

## Collections

It is possible for end-users to build and manage collections through an interactive interface, without having to work with computer files directly. The interface is designed to resemble standard installation packages that allow users to specify what they want done using a succession of menu pages. (Users have to have administrative privileges before they can create collections, however.) Facilities are provided to create a new collection, edit an existing one, rebuild a collection, and delete a collection.

## Logs

Finally, two kinds of logs can be examined: initialization logs and error logs. These are only really of interest to people maintaining the software. **Error logs are presently not working.**

## D.8 Translating the interface into other languages

The Greenstone software is designed to make it easy to present collections in a variety of different languages. At present, three languages are supported for the user interface—English, Maori, and Chinese. There is a Maori newspaper collection and a Chinese-language collection available from the NZDL home page, but in fact any collection can be viewed in all three languages: it is just a matter of making a choice on the *Preferences* page. This will shortly be altered to make it possible to specify which presentation languages are appropriate for each particular collection. When any particular text or icon is not available in the chosen language, the software is designed to default to the English-language equivalent.

New languages can be added by translating all items in the macro file *english.dm* and creating a new file, say *maori.dm*. It will be necessary to create new versions of all the textual icons as GIF images. The filenames for these are contained in *english.dm* and appropriate new filenames should be placed in the *.dm* file for the new language. In order to make the new *.dm* file active, all that is necessary is to place it in the *macros* directory and update the *macrofiles* line in the main configuration file *main.cfg* to include the new filename.