Greenstone
Funzionalità avanzate

Sommario

Digital Library Collections

- **Vi è una distinzione tra**
  - COSTRUIRE una collezione
  - FORNIRE informazioni agli utenti
- **È la stessa distinzione che esiste tra il ‘compile-time’ ed il ‘runtime’ nei linguaggi di programmazione**
- **La fase di costruzione è necessaria per preparare tutte le strutture dati che vengono poi utilizzate nella fase di delivery delle informazioni**

Costruzione manuale delle Collezioni

Costruzione di una collezione

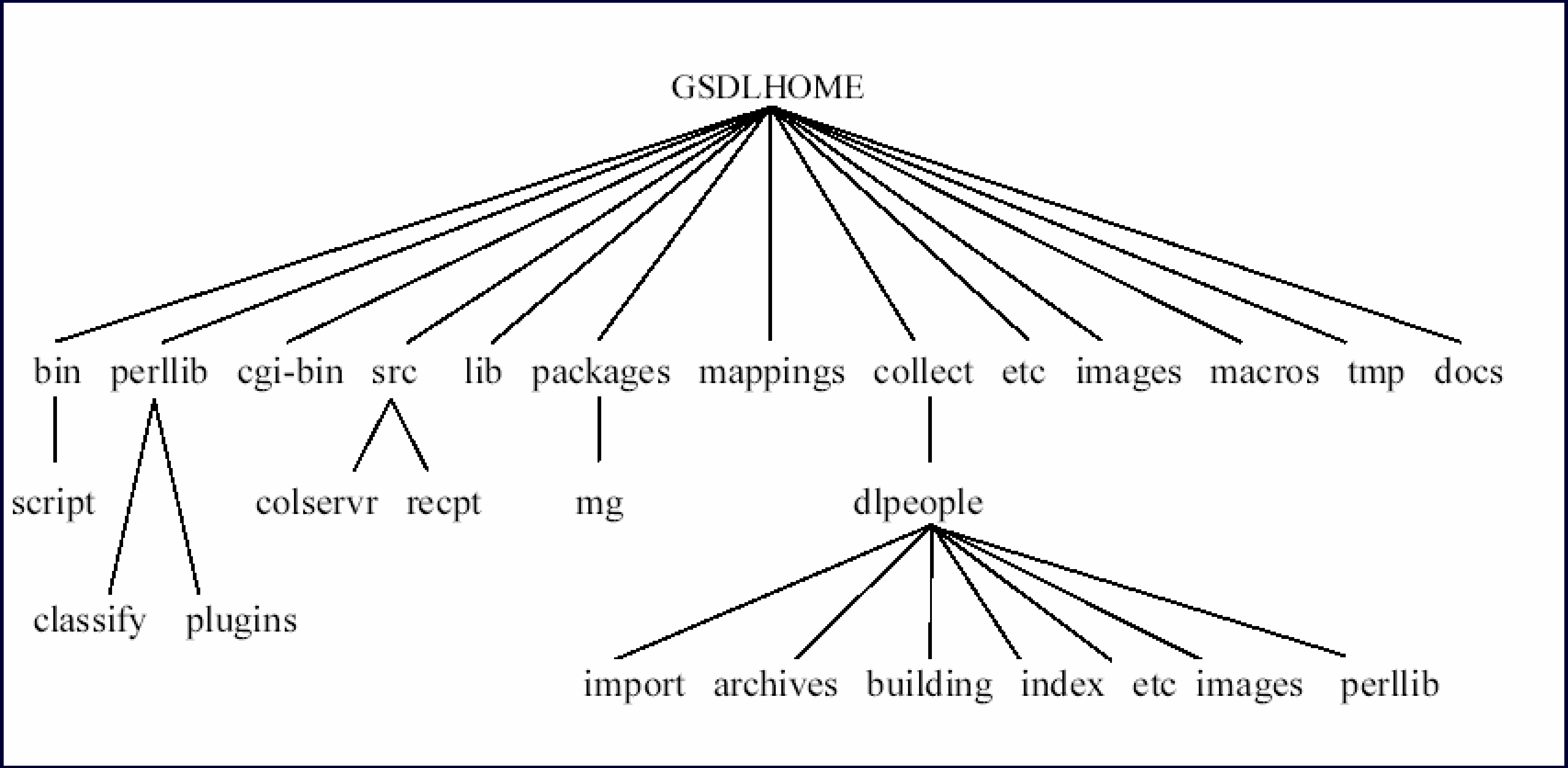
- Il processo che consiste nel prendere un insieme di documenti ed i metadati che li descrivono e creare tutti gli indici e le strutture dati che ne supportano la ricerca (search), il browsing, e la vualizzazione

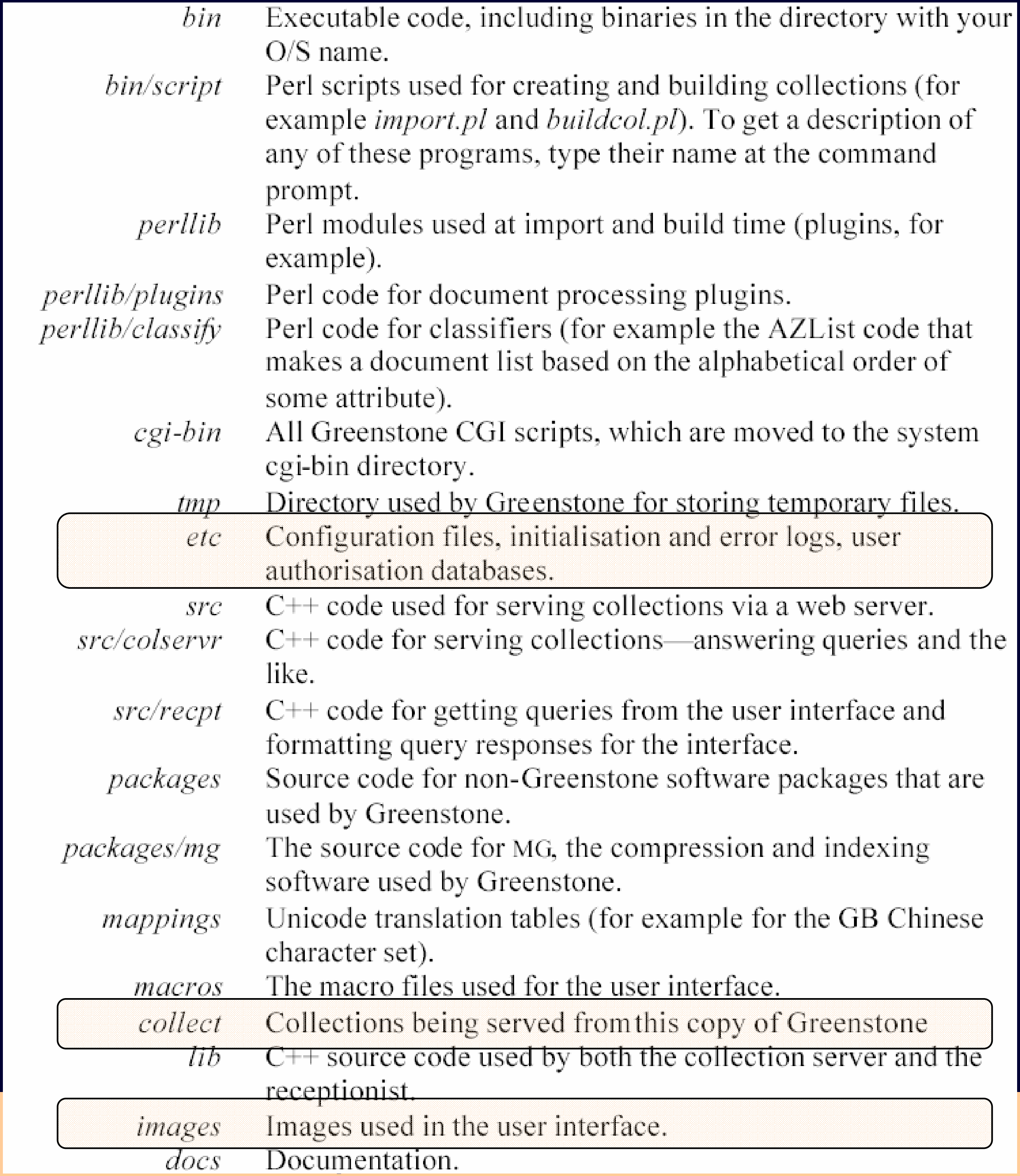
Costruzione di una collezione

- **La costruzione di una collezione prevede quattro fasi**
  - Make
    - **Creare uno scheletro di strutture e di file nel quale verranno inseriti I dati della collezione**
  - Import
    - **Importare I documenti ed I metadati e convertirli nel formato Greenstone**
  - Build
    - **Costruire gli indici e le strutture dati richieste**
  - Install
    - **Rendere operativa la collezione**

Make

- **Vengono create le sequenti directories**



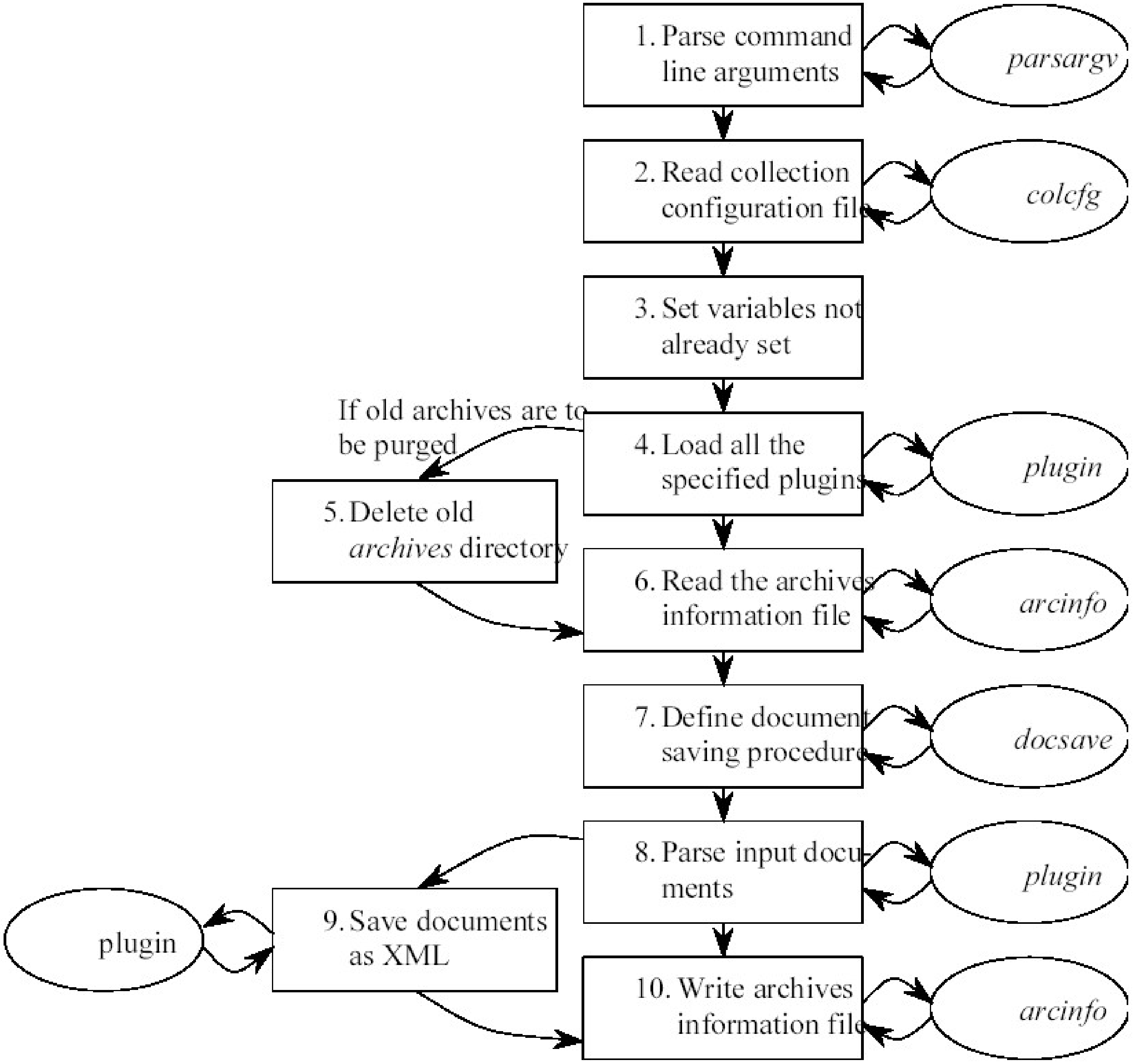


bin	Executable code, including binaries in the directory with your O/S name.
bin/script	Perl scripts used for creating and building collections (for example *import.pl* and *buildcol.pl*). To get a description of any of these programs, type their name at the command prompt.
perllib	Perl modules used at import and build time (plugins, for example).
perllib/plugins	Perl code for document processing plugins.
perllib/classify	Perl code for classifiers (for example the AZList code that makes a document list based on the alphabetical order of some attribute).
cgi-bin	All Greenstone CGI scripts, which are moved to the system cgi-bin directory.
tmp	Directory used by Greenstone for storing temporary files.
etc	Configuration files, initialisation and error logs, user authorisation databases.
src	C++ code used for serving collections via a web server.
src/colservr	C++ code for serving collections—answering queries and the like.
src/recpt	C++ code for getting queries from the user interface and formatting query responses for the interface.
packages	Source code for non-Greenstone software packages that are used by Greenstone.
packages/mg	The source code for MG, the compression and indexing software used by Greenstone.
mappings	Unicode translation tables (for example for the GB Chinese character set).
macros	The macro files used for the user interface.
collect	Collections being served from this copy of Greenstone
lib	C++ source code used by both the collection server and the receptionist.
images	Images used in the user interface.
docs	Documentation.

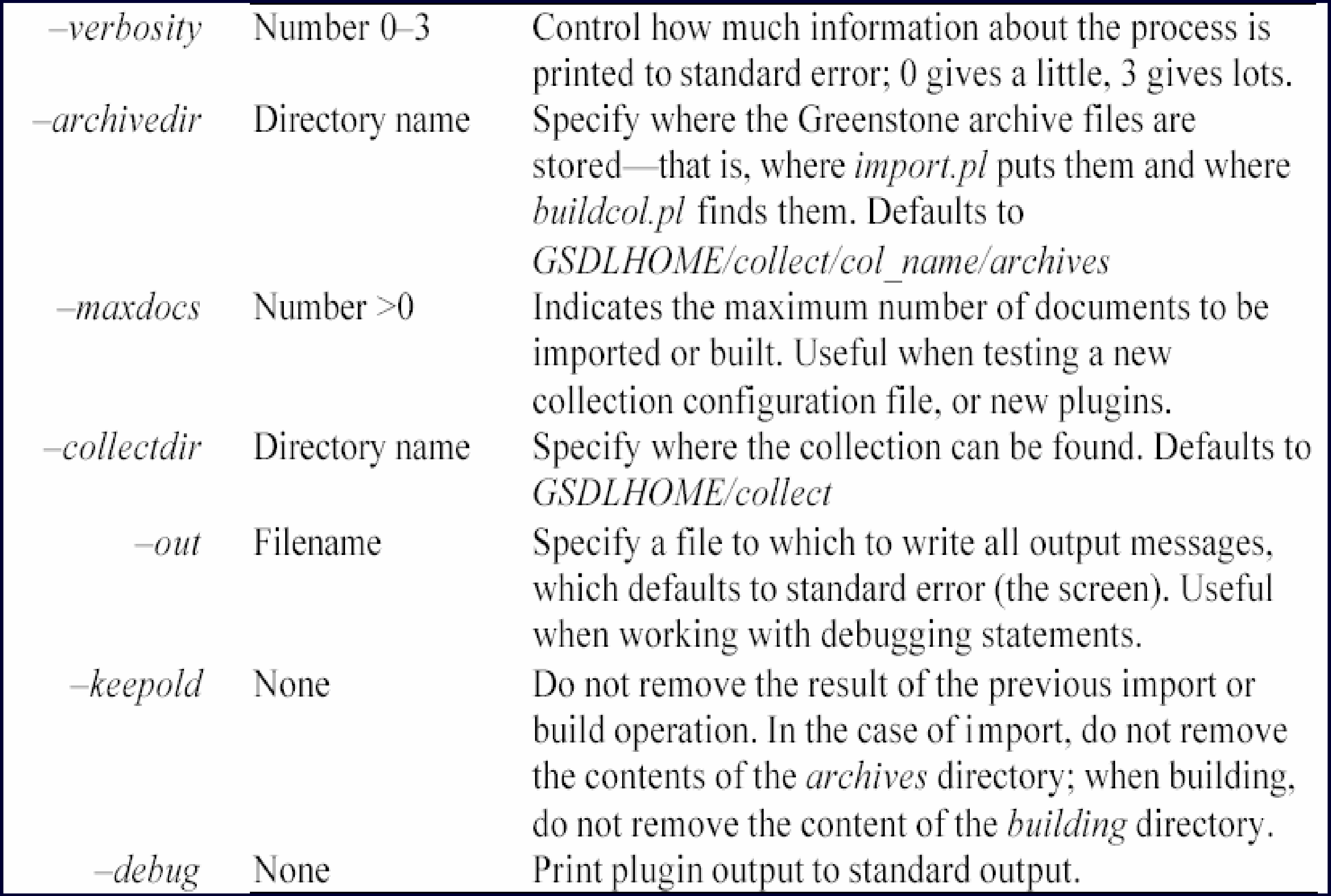
Import Process

- **Brings documents and metadata into the system in a standardized XML form**
- **Original material placed in *import* directory**
- **Import process transforms it to files in the *archives* directory**
- **The original material can be deleted**
  - Collection can be rebuilt from archive files
- **New material added to collection by placing it in *import* directory and re-executing the import process**
  - The new material finds it way into archives along with existing files
- **To keep the source form of collections**
  - Do not delete the archives
  - “Source” form can be augmented and rebuilt later

The Import Process

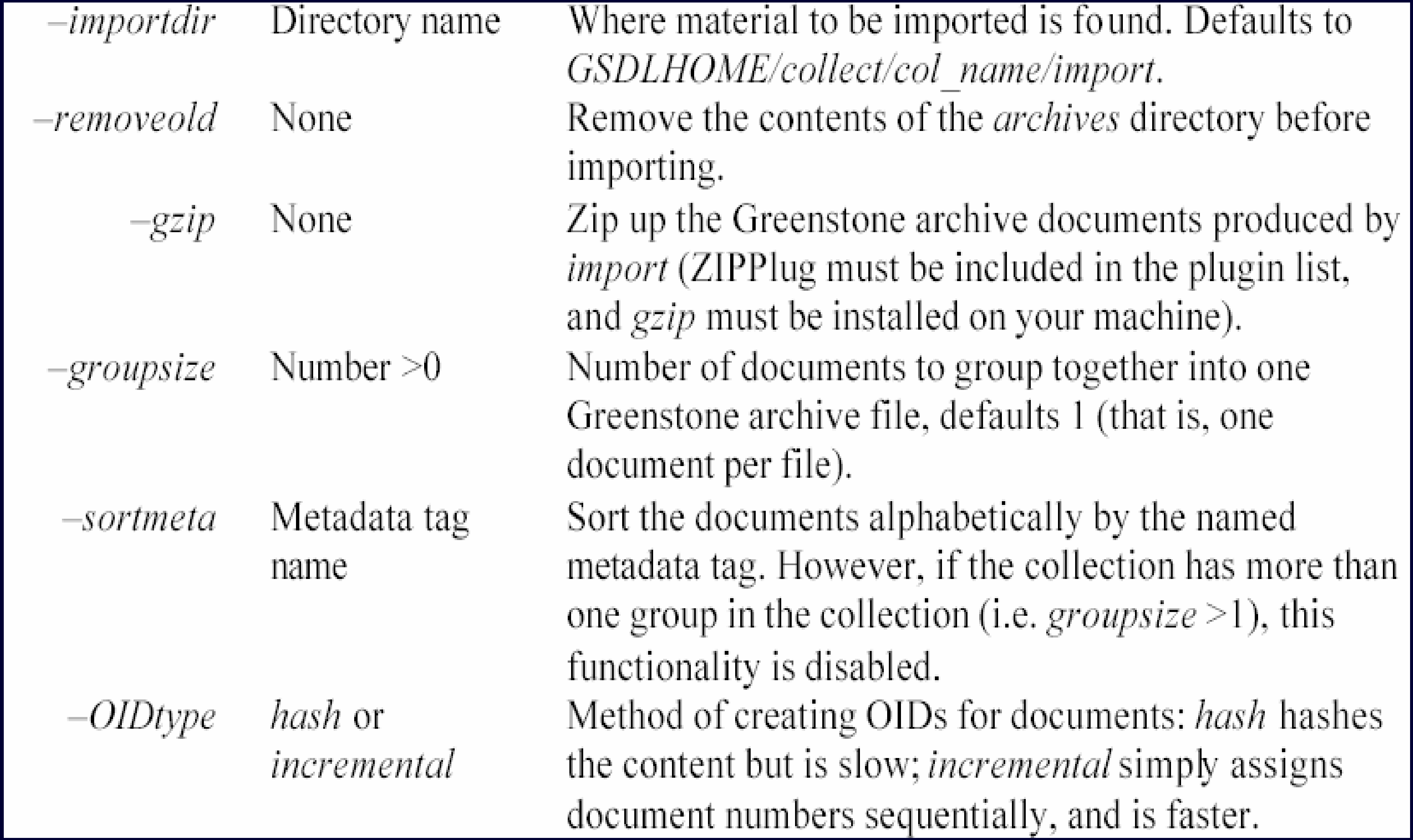

1. Parse command line arguments
parsargv
2. Read collection configuration file
colcfg
3. Set variables not already set
If old archives are to be purged
4. Load all the specified plugins
plugin
5. Delete old *archives* directory
6. Read the archives information file
arcinfo
7. Define document saving procedure
docsave
8. Parse input documents
plugin
9. Save documents as XML
plugin
10. Write archives information file
arcinfo

Options for Import



–verbosity	Number 0–3	Control how much information about the process is printed to standard error; 0 gives a little, 3 gives lots.
–archivedir	Directory name	Specify where the Greenstone archive files are stored—that is, where *import.pl* puts them and where *buildcol.pl* finds them. Defaults to *GSDLHOME/collect/col_name/archives*
–maxdocs	Number >0	Indicates the maximum number of documents to be imported or built. Useful when testing a new collection configuration file, or new plugins.
–collectdir	Directory name	Specify where the collection can be found. Defaults to *GSDLHOME/collect*
–out	Filename	Specify a file to which to write all output messages, which defaults to standard error (the screen). Useful when working with debugging statements.
–keepold	None	Do not remove the result of the previous import or build operation. In the case of import, do not remove the contents of the *archives* directory; when building, do not remove the content of the *building* directory.
–debug	None	Print plugin output to standard output.

Additional Options for Import



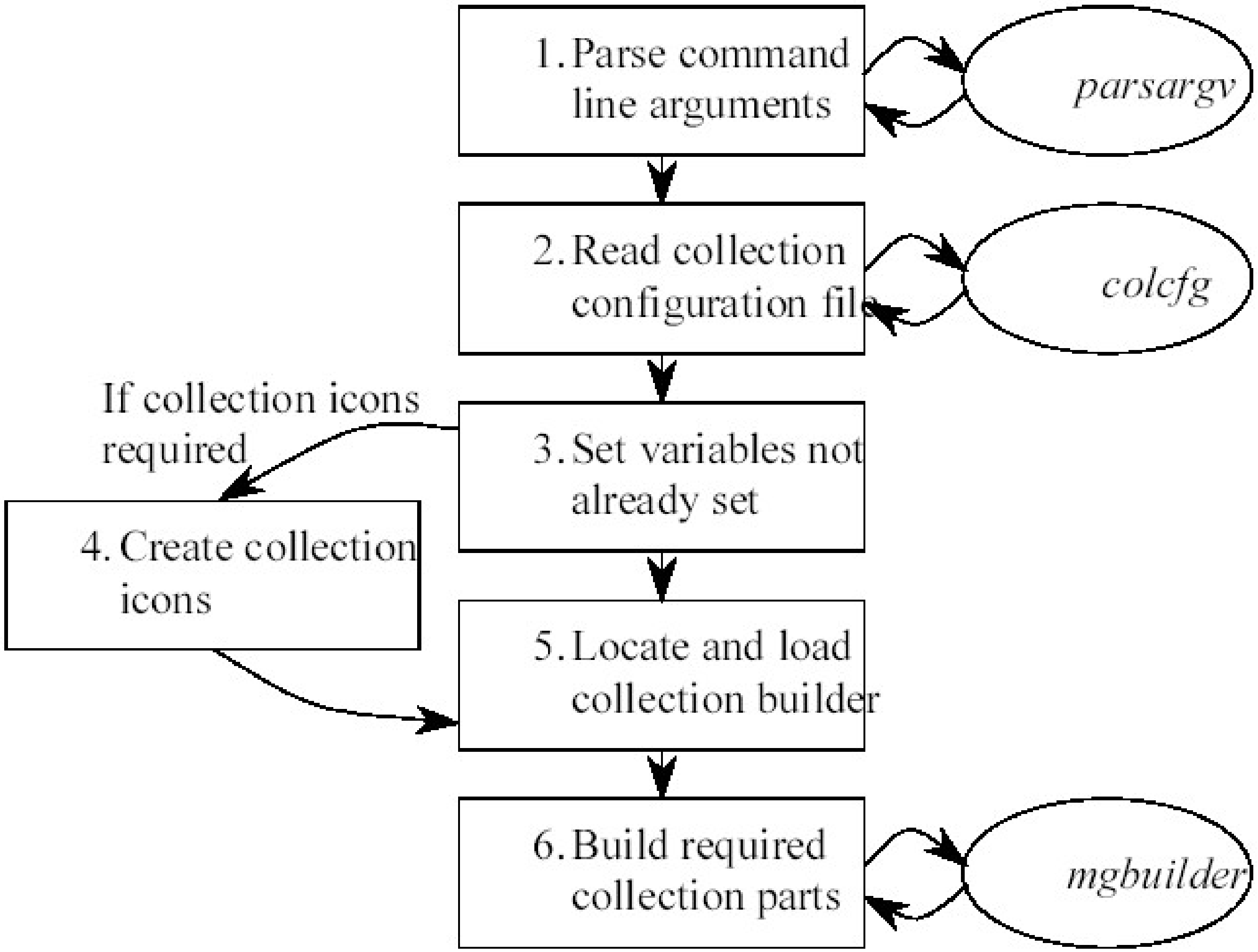
–importdir	Directory name	Where material to be imported is found. Defaults to *GSDLHOME/collect/col_name/import*.
–removeold	None	Remove the contents of the *archives* directory before importing.
–gzip	None	Zip up the Greenstone archive documents produced by *import* (ZIPPlug must be included in the plugin list, and *gzip* must be installed on your machine).
–groupsize	Number >0	Number of documents to group together into one Greenstone archive file, defaults 1 (that is, one document per file).
–sortmeta	Metadata tag name	Sort the documents alphabetically by the named metadata tag. However, if the collection has more than one group in the collection (i.e. *groupsize* >1), this functionality is disabled.
–OIDtype	*hash* or *incremental*	Method of creating OIDs for documents: *hash* hashes the content but is slow; *incremental* simply assigns document numbers sequentially, and is faster.

The Build Process

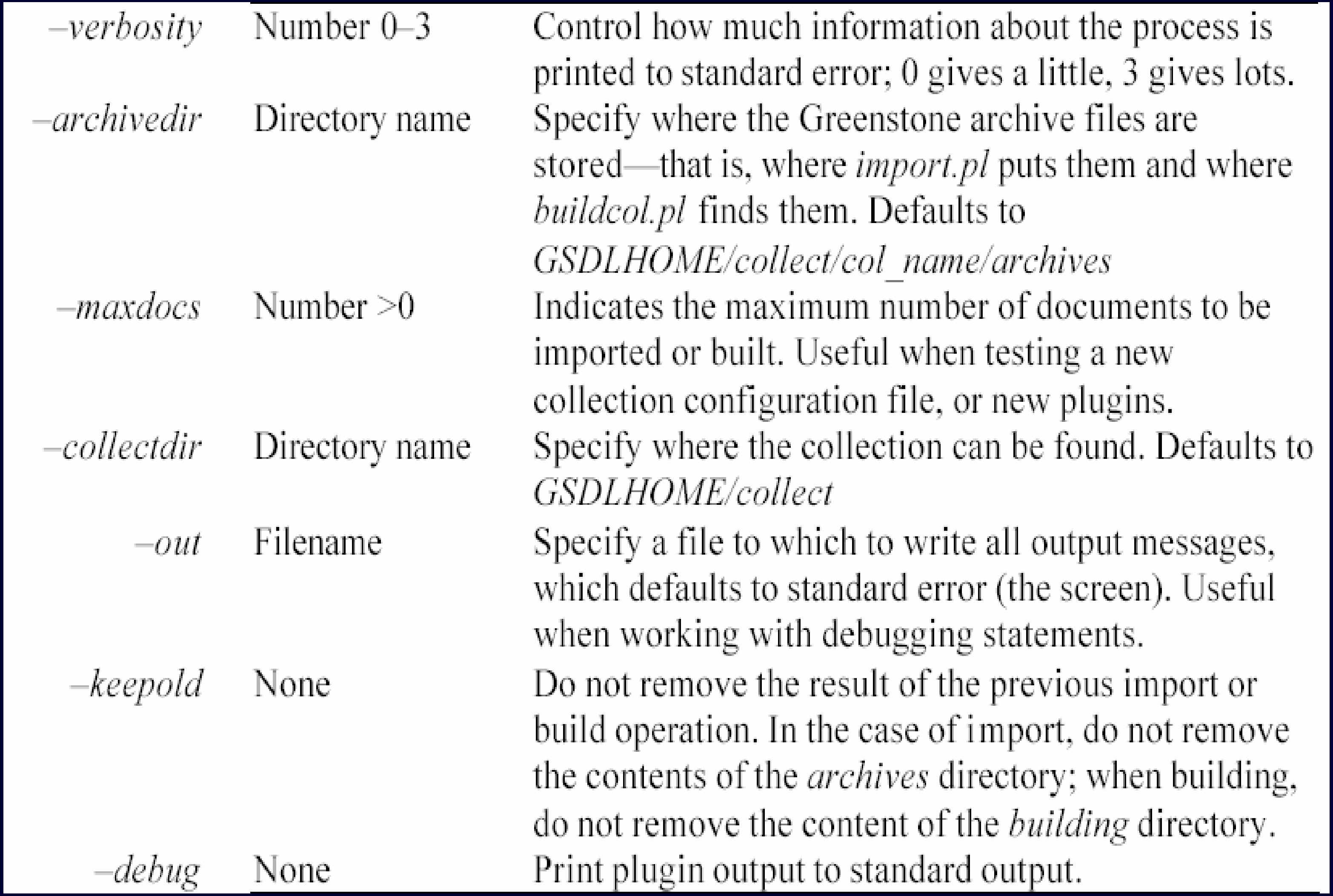
- **Creates the indexes and data structures that make the collection operational**

- **Indexes for the whole collection are build all at once**
  - Build process does not work incrementally
  - Adding new material to *archives* requires that entire collection be rebuilt (by issuing *buildcol.pl*)
  - Most collections can be rebuilt overnight

The Build Process

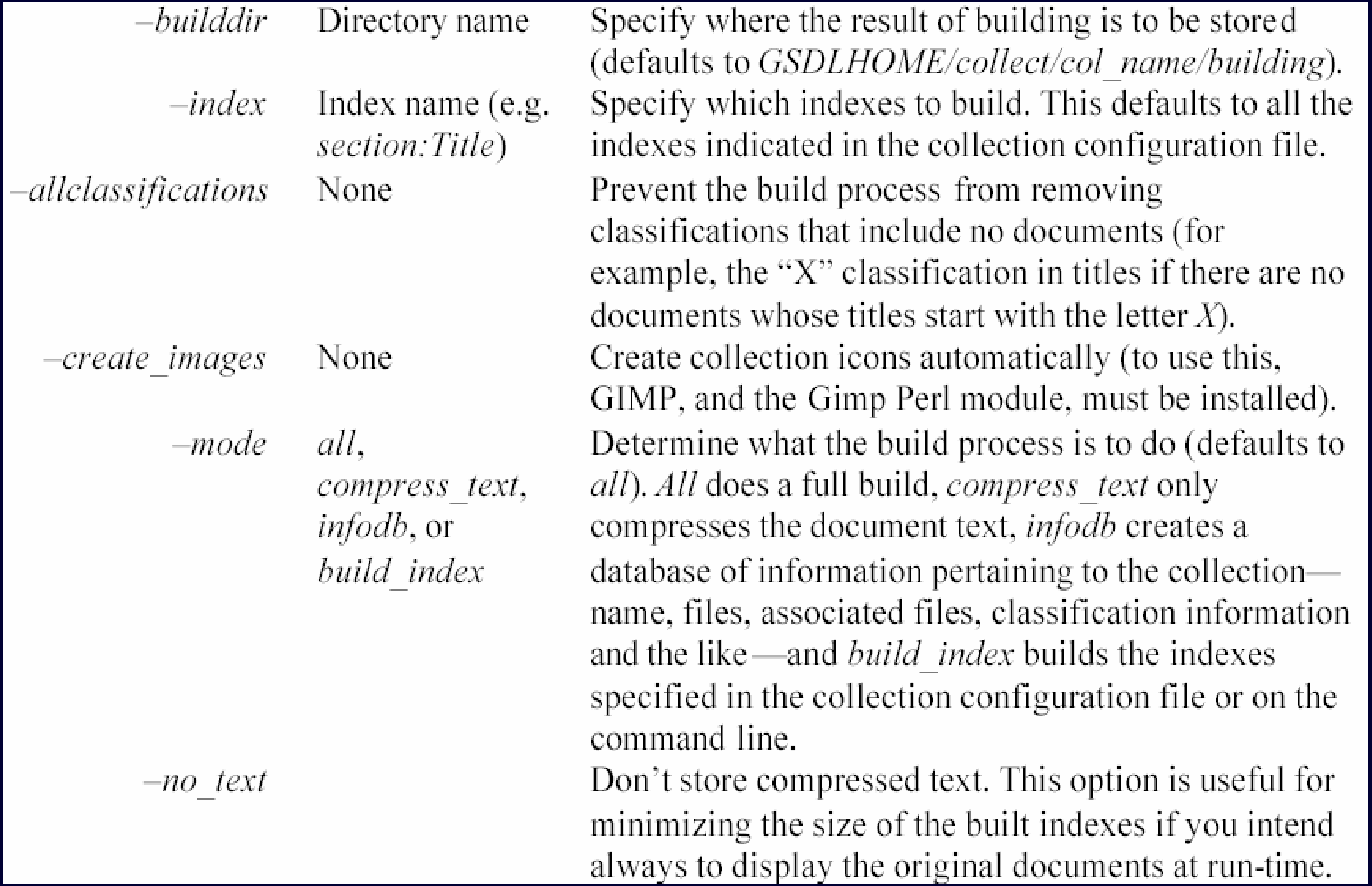

1. Parse command line arguments
parsargv
2. Read collection configuration file
colcfg
If collection icons required
3. Set variables not already set
4. Create collection icons
5. Locate and load collection builder
6. Build required collection parts
mgbuilder

Options for Build



–verbosity	Number 0–3	Control how much information about the process is printed to standard error; 0 gives a little, 3 gives lots.
–archivedir	Directory name	Specify where the Greenstone archive files are stored—that is, where *import.pl* puts them and where *buildcol.pl* finds them. Defaults to *GSDLHOME/collect/col_name/archives*
–maxdocs	Number >0	Indicates the maximum number of documents to be imported or built. Useful when testing a new collection configuration file, or new plugins.
–collectdir	Directory name	Specify where the collection can be found. Defaults to *GSDLHOME/collect*
–out	Filename	Specify a file to which to write all output messages, which defaults to standard error (the screen). Useful when working with debugging statements.
–keepold	None	Do not remove the result of the previous import or build operation. In the case of import, do not remove the contents of the *archives* directory; when building, do not remove the content of the *building* directory.
–debug	None	Print plugin output to standard output.

Additional Options for Build



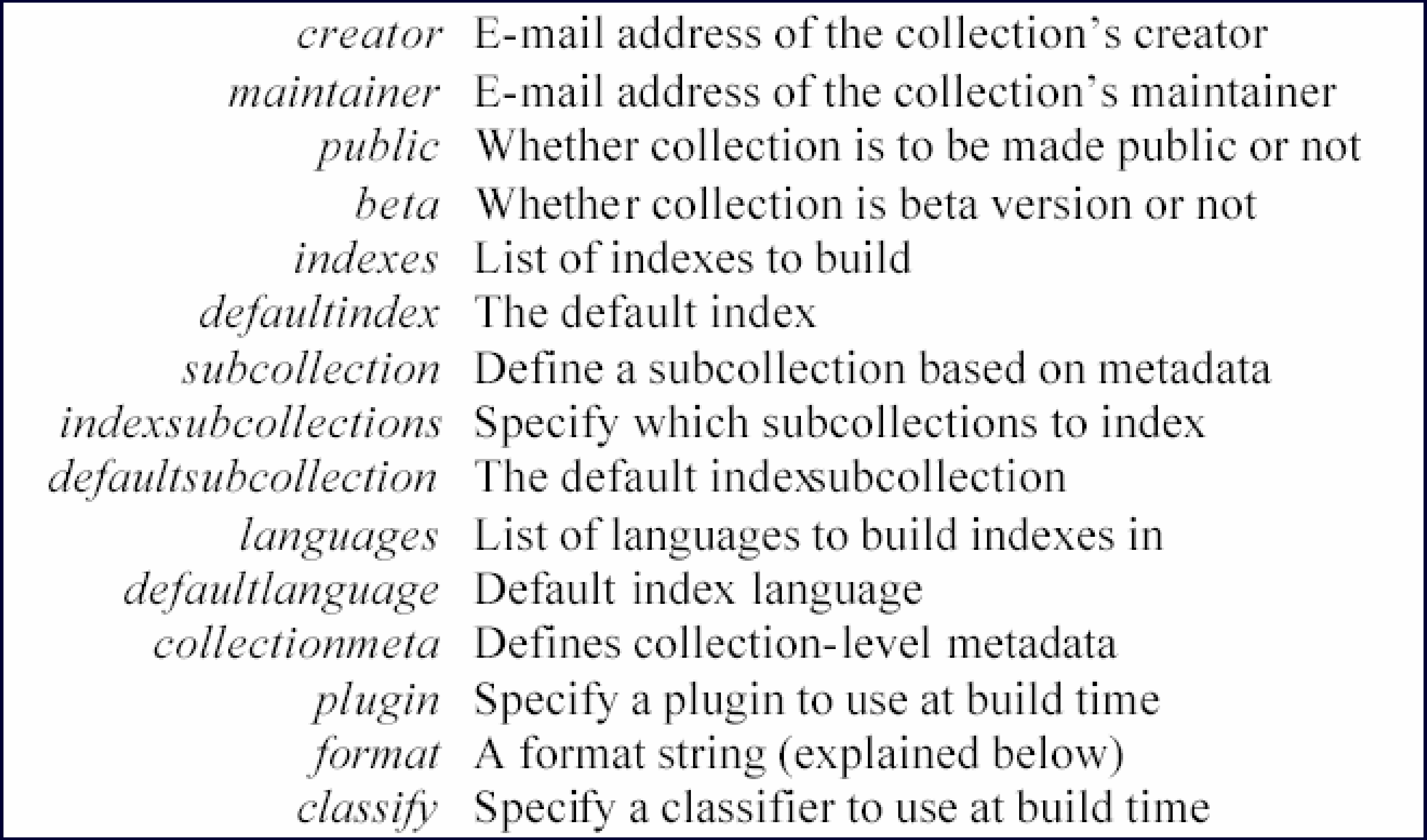
–builddir	Directory name	Specify where the result of building is to be stored (defaults to *GSDLHOME/collect/col_name/building*).
–index	Index name (e.g. *section:Title*)	Specify which indexes to build. This defaults to all the indexes indicated in the collection configuration file.
–allclassifications	None	Prevent the build process from removing classifications that include no documents (for example, the "X" classification in titles if there are no documents whose titles start with the letter *X*).
–create_images	None	Create collection icons automatically (to use this, GIMP, and the Gimp Perl module, must be installed).
–mode	*all*, *compress_text*, *infodb*, or *build_index*	Determine what the build process is to do (defaults to *all*). *All* does a full build, *compress_text* only compresses the document text, *infodb* creates a database of information pertaining to the collection—name, files, associated files, classification information and the like—and *build_index* builds the indexes specified in the collection configuration file or on the command line.
–no_text		Don't store compressed text. This option is useful for minimizing the size of the built indexes if you intend always to display the original documents at run-time.

Collection Configuration File

Collection Configuration File

- **Collection Configuration File**
  - Defines the structure of a collection
  - Governs how the collection is built
  - Specifies how the collection will appear to users
- **Ogni linea del Collection Configuration File è una coppia “attributo”, “valore”**

Collection Configuration File [1/4]



creator	E-mail address of the collection's creator
maintainer	E-mail 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
subcollection	Define a subcollection based on metadata
indexsubcollections	Specify which subcollections to index
defaultsubcollection	The default indexsubcollection
languages	List of languages to build indexes in
defaultlanguage	Default index language
collectionmeta	Defines collection-level metadata
plugin	Specify a plugin to use at build time
format	A format string (explained below)
classify	Specify a classifier to use at build time

Collection configuration file [2/4]

Indici creati durante il build della collezione

Plugin da usare per convertire documenti nel formato Greenstone

Classificatore per creare una lista alfabetica di titoli

```
creator         username@email.com
maintainer      username@email.com
public          true
beta            false

indexes         section:text section:Title document:text

plugin          GAPlug
plugin          HTMLPlug -description_tags -cover_image
plugin          WordPlug -description_tags
plugin          ArcPlug
plugin          RecPlug -show_progress -use_metadata_files

classify        AZList metadata Title
```

Collection configuration file [3/4]

```
format DocumentText "<h3>[Title]</h3>\\n\\n<p>[Text]"
format DocumentImages true
format DocumentButtons "Expand Text|Expand
    Contents|Detach|Highlight"
```

Formato di presentazione dei metadati

Metadati della collezione

```
Collectionmeta collectionname "greenstone demo"
Collectionmeta collectionextra "This is a
    demonstration collection”
Collectionmeta iconcollection
    "_httpprefix_/collect/demo/images/img.gif"
```

Collection configuration file [4/4]

```
Collectionmeta collectionextra “collection description"
Collectionmeta collectionextra "This is a demonstration
   collection”
```

Descrive la collezione. Viene usato come testo nella sezione “About this collection”

```
Collectionmeta iconcollection
   "_httpprefix_/collect/demo/images/img.gif"
```

Immagine che descrive
la collezione. Viene usata nella
home page della collezione

Subcollections [1/2]

- Greenstone permette di costruire sotto-collezioni, e di costruire indici per ognuna di esse.
- Consideriamo una collezione costituita documenti testuali, alcuni tratti dal “Journal of Digital Libraries” ed altri no
- Vogliamo creare due sotto-collezioni ed indici al livello di section

```
indexes      section:text
subcollection dl “Title/^Journal of Digital Libraries/i”
subcollection other “!Title/^Journal of Digital
   Libraries/i”
indexsubcollections dl other dl,other
```

Subcollections [2/2]

- Lo stesso meccanismo può essere utilizzato per creare indici per collezioni che contengono documenti in diverse lingue
- La lingua del documento è un metadato (en per l'inglese, it per italiano, ecc.)

```
indexes     section:text section:Title document:text
Languages it en fr
```

- Vengono creati indici separati per section text, section title, e document text per le tre diverse lingue (9 indici in totale)

Cross-collection searching

- In Greenstone è possibile effettuare ricerche su più collezioni, come se fossero costituite da una sola collezione
- Questa funzionalità viene abilitata inserendo nel Collection Configuration File

```
supercollection  col_1 col_2 …
```

- Nel caso che le collezioni siano denominate col_1, col_2, ecc.
- Questa indicazione deve essere presente nel file di configurazione di tutte le collezioni coinvolte.

Plug-ins

Plug-ins

- **I plug-in sono moduli software che gestiscono**
  - Conversioni di formato
  - Estrazione di metadati
- **I plug-in permettono di estendere le funzionalità di Greenstone**
  - È possibile sviluppare nuovi plug-in per estendere i tipi di documenti gestiti o i metadati che possono essere estratti
- **I plug-in sono scritti nel linguaggio Perl. Sono tutti derivati da un plug-in base: *BasPlug.***
- ***BasPlug* crea un nuovo documento archivio di Greenstone ed assegna un identificatore al documento**
- **Maggiori informazioni su ogni plug-in si possono avere digitando “perl – S pluginfo.pl nome-plugin” alla linea comandi di windows**

Plug-Ins

- **Plug-ins do most of the work of the import process**
- **Operate in the order in which they are listed in the *collect.cfg* file**
  - Input file is passed to each plug-in until one is found that can process it
- **If there is no plug-in that can process a file, a warning is printed**
- **Plug-ins determine the traversal of the subdirectory structure in the import directory**
  - *RecPlug* - processes directories, recurses through directory structures and passes the name through the plug-in list
  - *GAPlug* – processes Greenstone Archive Format documents (in the archives directory structure)
  - *ArcPlug* – used during building, processes list of document OIDs produced during import (list is stored in *archives.inf* file)

Plug-ins & Document Formats

- **Plug-ins are specified in the collection configuration file**
- **File name determines document format**
- **Widely used document formats:**

TEXTPlug	PSPlug	SRCPlug
HTMLPlug	EMAILPlug	ImagePlug
WORDPlug	BibTexPlug	ZIPPlug
PDFPlug	ReferPlug	

Text Files

- **TEXTPlug Plug-In**
  - *.txt
  - *.text
- **Plain text file**
- **Title metadata based on the first line of the file**

HTML Files

- **HTMLPlug Plug-In**
  - *.htm
  - *.html
  - .shtml
  - .shm
  - .asp
  - .php
  - .cgi

HTML Files

- **HTMLPlug Plug-In**
  - Imports HTML files
  - Title metadata extracted from the HTML <title> tag
  - Other HTML <meta> tag data can be extracted
  - Parses and processes any links in the file
  - Links to other files in the collection are trapped and replaced by references to the document

HTML Files

- ***file_is_url***
  - Optional switch within the HTML plug-in
  - Causes URL metadata to be inserted into each document, based on the file-name convention that is adopted by the mirroring package. The collection uses this metadata to allow readers to refer to the original source material rather than a local copy

Microsoft Word Files

- **WORDPlug Plug-In**
  - *.doc
- **Imports Microsoft Word documents**
- **Greenstone uses independent programs to convert Word files to HTML**
  - Many variants on the Word format
  - Older Word formats use a simple text string extraction

PDF Files

- **PDFPlug Plug-In**
  - *.pdf
- **Imports PDF Files**
- **Adobe’s Portable Document Format**
- **Greenstone uses independent programs to convert PDF files to HTML**

PostScript Files

- **PSPlug Plug-In**
  - *.ps
- **Imports PostScript Files**
- **Works best when a standard conversion program is already installed on the computer**
- **Uses simple text extraction algorithm if no conversion program is present**

Email Files

- **EMAILPlug**
  - *.email
- **Imports files containing email**
  - Each source is checked for e-mail contents
- **Extracts metadata:**
  - Subject
  - To
  - From
  - Date
- **Deals with common formats**
  - Netscape, Eudora, Unix mail readers

Compressed & Archived Files

- **ZIPPlug Plug-In**
  - *.zip
  - *.tar
  - .gz
  - *.z
  - *.tgz
  - *.bz
- **Relies on standard utility programs being present**

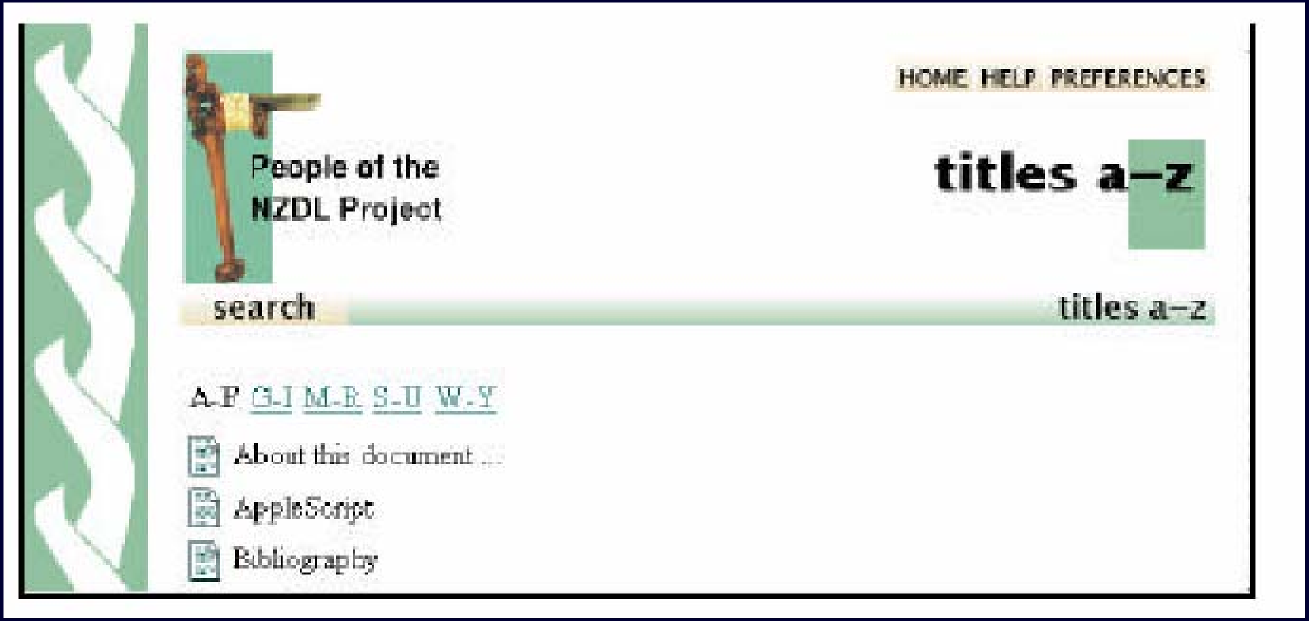
Classifiers

I Classifiers

- Gestiscono strutture per il browsing della collezione
- Vengono specificati nel Collection Configuration File
- Per ogni classifier vi è una linea del tipo
  - `classify nome_classifier opzioni`
- I programmatori possono scrivere nuovi classifiers per creare nuove strutture di browsing

Esempi di Classifier [1/4]

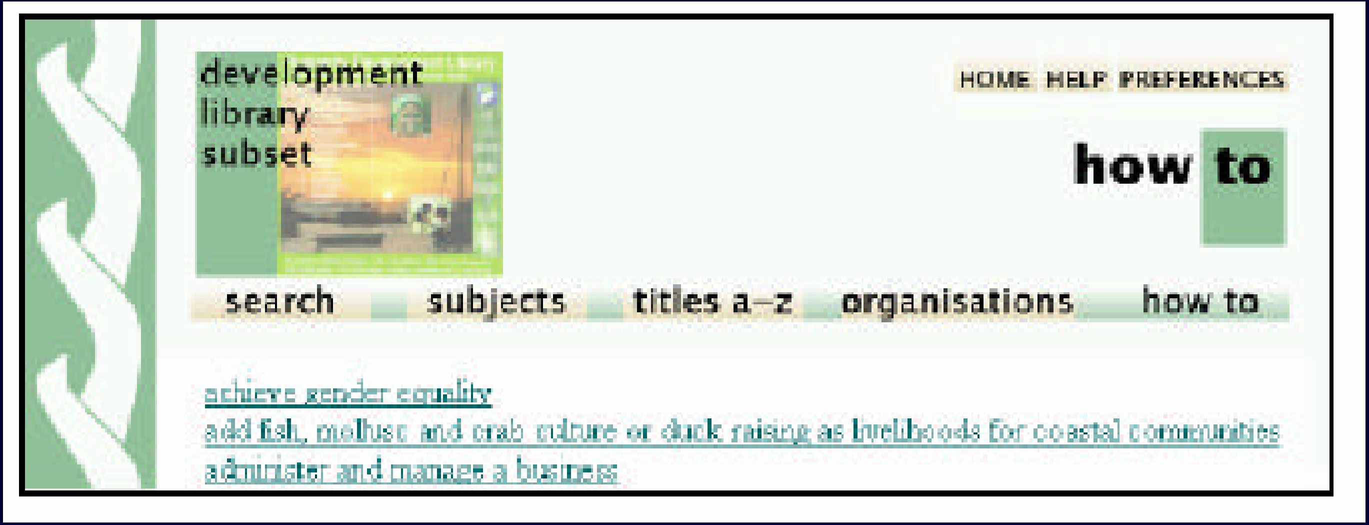
- **AZList classifier**
  - Crea una lista ordinata alfabeticamente di elementi
  - Ad es. `Classify AZList -metadata Title`



About this document ...
AppleScript
Bibliography

Esempi di Classifier [2/4]

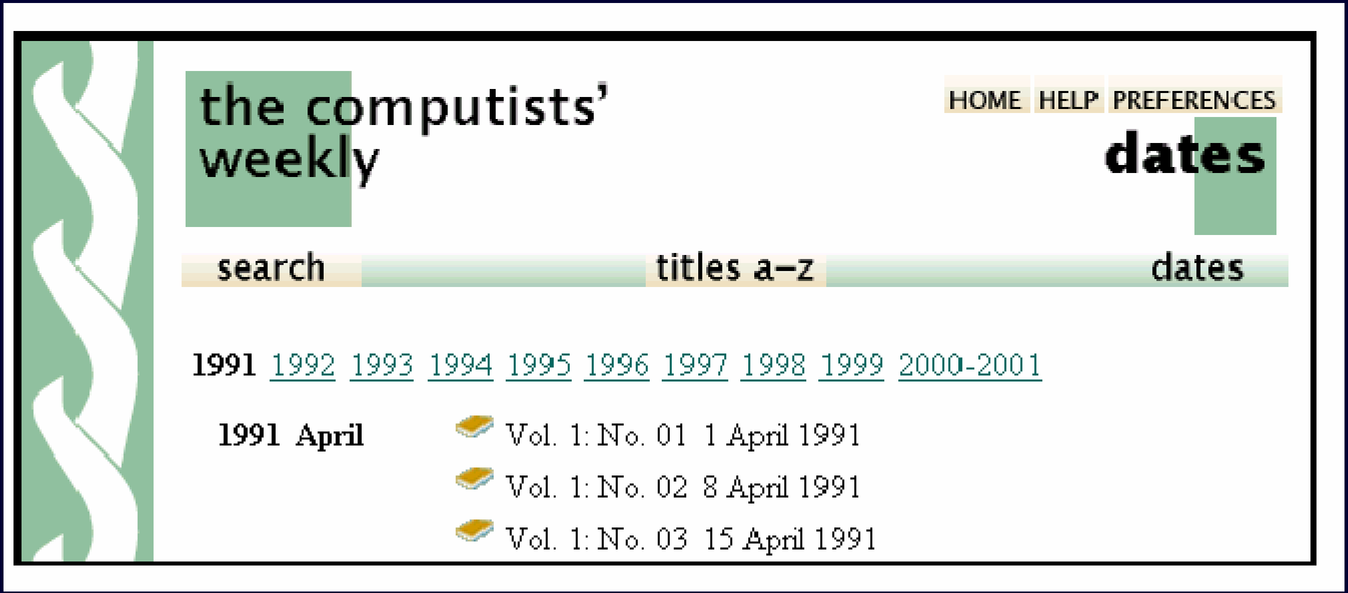
- **List classifier**
  - Crea una lista ordinata di elementi e li visualizza senza alcun ordine specifico
  - Ad es. `classify List –metadata Howto`



achieve gender equality
add fish, mollusc and crab culture or duck raising as livelihoods for coastal communities
administer and manage a business

Esempi di Classifier [3/4]

- **DateList classifier**
  - Crea una lista ordinata di elementi data
  - Ad es. `classify DateList -metadata date`

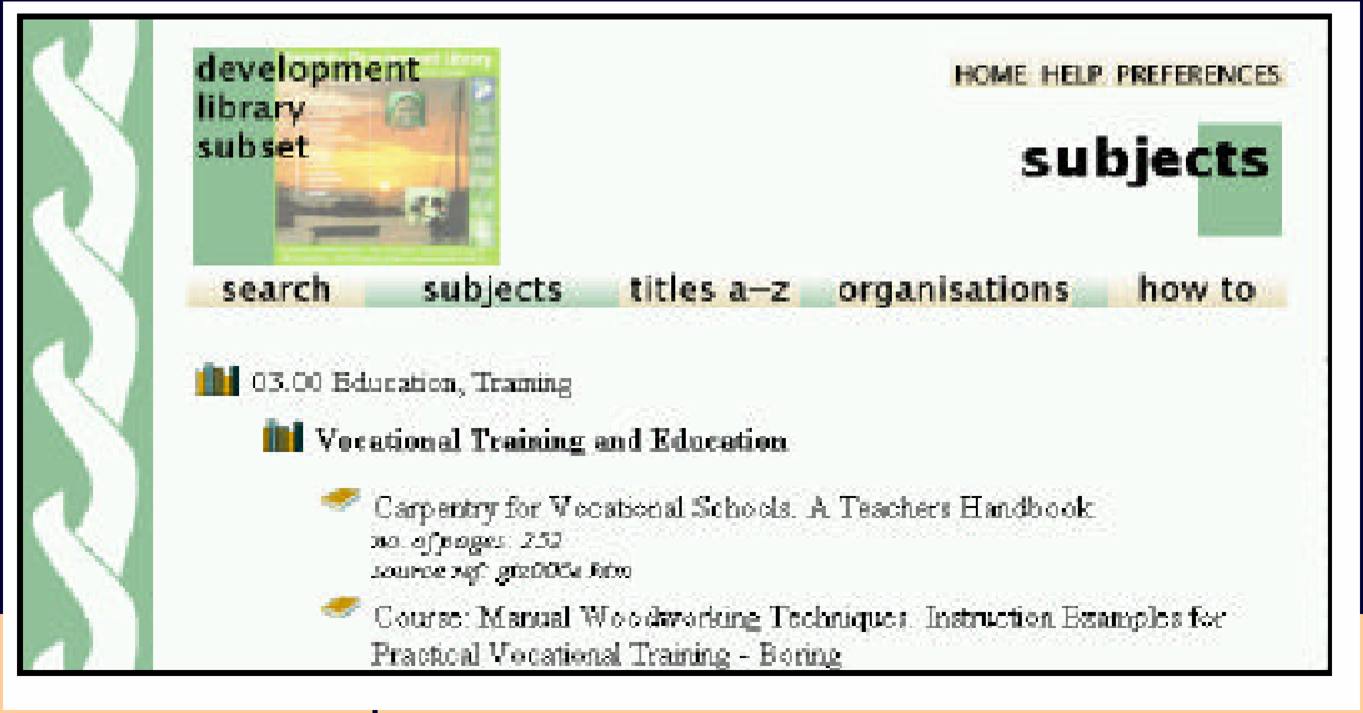


1991 1992 1993 1994 1995 1996 1997 1998 1999 2000-2001

1991 April
Vol. 1: No. 01 1 April 1991
Vol. 1: No. 02 8 April 1991
Vol. 1: No. 03 15 April 1991

Esempi di Classifier [4/4]

- **Classifier gerarchici**
  - Creano classificazioni gerarchiche e sono utili per la classificazione di soggetti ed organizzazioni
  - Ad es. `classify Hierarchy -hfile sub.txt -metadata Subject -sort Title`



03.00 Education, Training

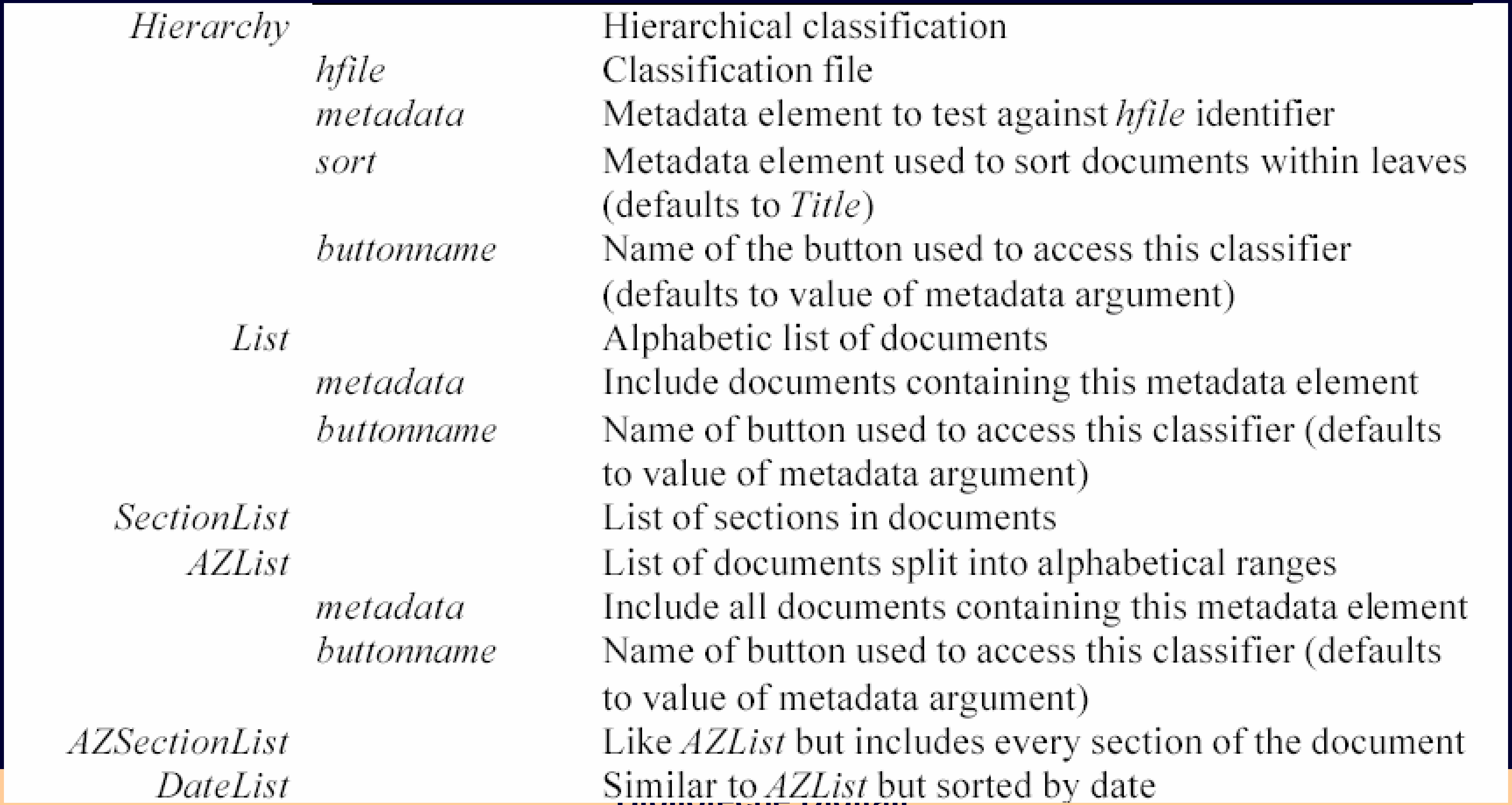
Vocational Training and Education

Carpentry for Vocational Schools. A Teachers Handbook
no. of pages: 252
source ref: gtz005e.htm

Course: Manual Woodworking Techniques. Instruction Examples for Practical Vocational Training - Boring

I classifiers

- **Informazioni sui classifiers si possono avere digitando dalla linea comandi**
  - `perl – S classinfo.pl nome-classifier`



Hierarchy		Hierarchical classification
	hfile	Classification file
	metadata	Metadata element to test against *hfile* identifier
	sort	Metadata element used to sort documents within leaves (defaults to *Title*)
	buttonname	Name of the button used to access this classifier (defaults to value of metadata argument)
List		Alphabetic list of documents
	metadata	Include documents containing this metadata element
	buttonname	Name of button used to access this classifier (defaults to value of metadata argument)
SectionList		List of sections in documents
AZList		List of documents split into alphabetical ranges
	metadata	Include all documents containing this metadata element
	buttonname	Name of button used to access this classifier (defaults to value of metadata argument)
AZSectionList		Like *AZList* but includes every section of the document
DateList		Similar to *AZList* but sorted by date

Indexes

Searching Involves Indexes

- **Searching is provided by indexes built from different parts of the documents**
  - Entire documents
  - Paragraphs
  - Titles
  - Sections
  - Section headings
  - Figure captions

Indexes

- **Indexes can be created automatically using**
  - Documents
  - Supporting files
- **Indexes can be rebuilt automatically**
  - New document in the same format becomes available
  - Process can awake, check for new material, and rebuild the indexes

Plug-ins for Indexing

- **Source documents are converted into standard XML form for indexing using plug-ins**
- **Standard plug-ins process**
  - Plain text
  - HTML
  - Word
  - PDF
  - Usenet and email messages
- **New plug-ins can be written for other document types**

Search Terms

- **Search Terms in Greenstone:**
  - Alphabetic characters
  - Digits
- **Separated by white space**
  - Punctuation acts as white space

Two Types of Queries

- **Query for ALL of the words**
  - Boolean AND
- **Query for SOME of the words**
  - Ranked

Indexes to Search

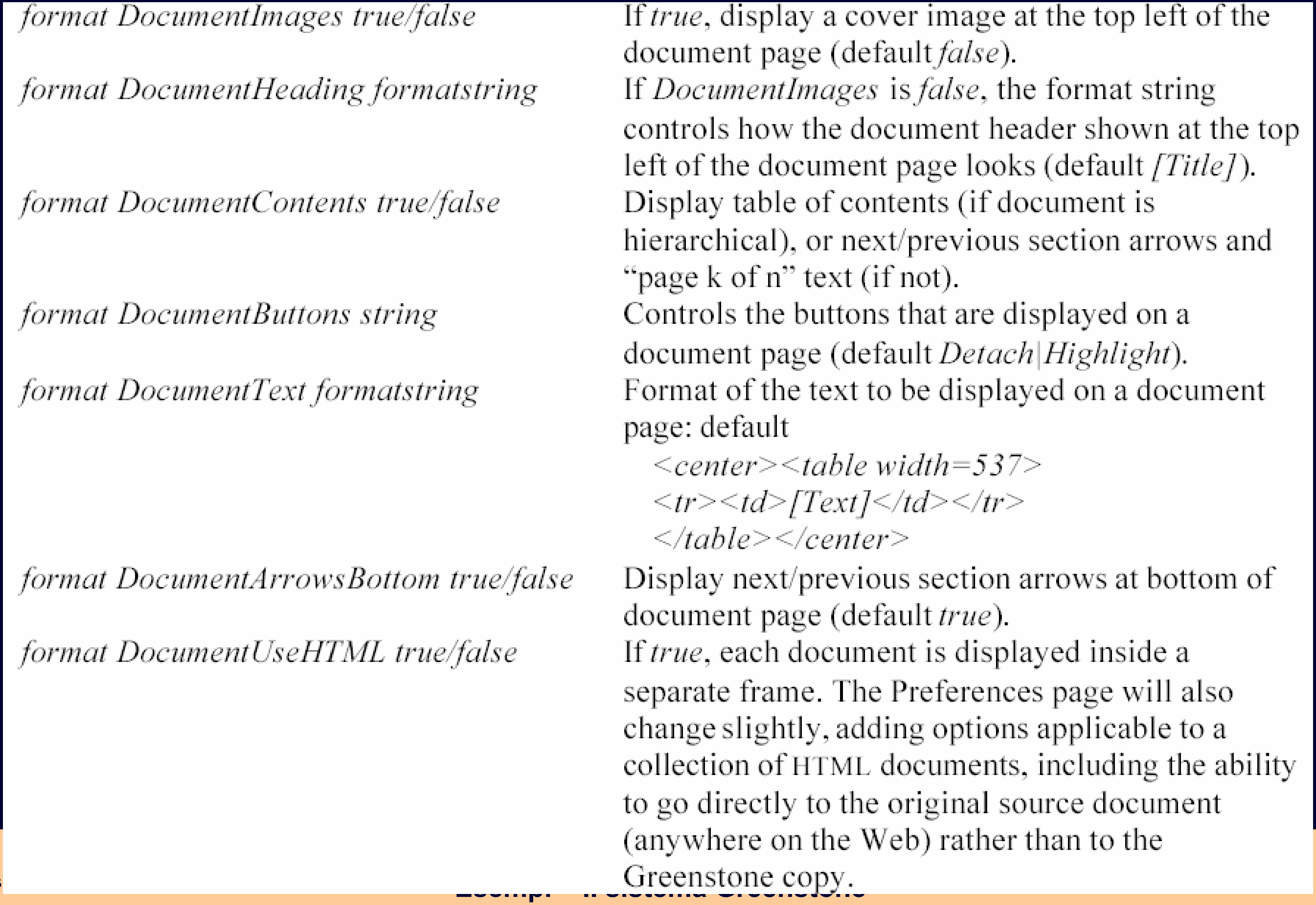
- In most collections, you can choose different indexes to search
- Examples:
  - Author and title indexes
  - Chapter and paragraph indexes
- Usually the full matching document is returned regardless of index searched

Come formattare l’output

Introduzione

- **Le pagine web visualizzate da Greenstone non sono preesistenti ma vengono generate**
- **Le modalità di visualizzazione sono controllate dal comando "format" del Collection Configuration File**
- **Elementi della pagina controllabili**
  - Item della pagina che presentano i documenti
  - Liste prodotte dai classifiers e risultati delle ricerche

Visualizzazione degli item nella pagina



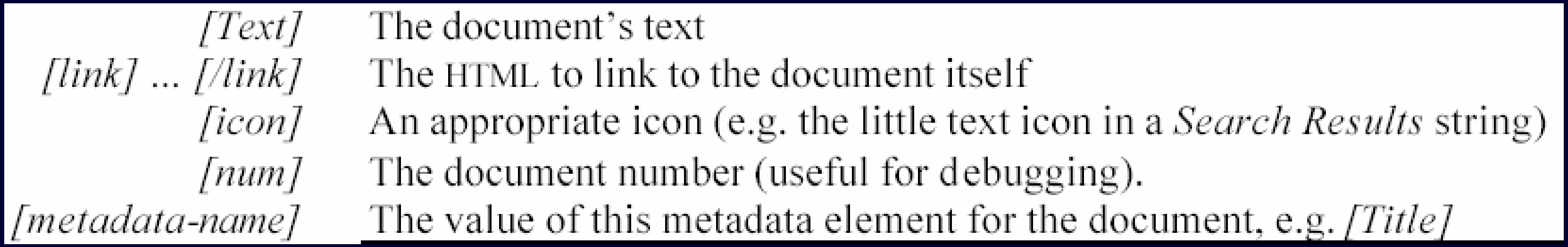
format DocumentImages true/false	If *true*, display a cover image at the top left of the document page (default *false*).
format DocumentHeading formatstring	If *DocumentImages* is *false*, the format string controls how the document header shown at the top left of the document page looks (default *[Title]*).
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 *Detach*|*Highlight*).
format DocumentText formatstring	Format of the text to be displayed on a document page: default *<center><table width=537> <tr><td>[Text]</td></tr> </table></center>*
format DocumentArrowsBottom true/false	Display next/previous section arrows at bottom of document page (default *true*).
format DocumentUseHTML true/false	If *true*, each document is displayed inside a separate frame. The Preferences page will also change slightly, adding options applicable to a collection of HTML documents, including the ability to go directly to the original source document (anywhere on the Web) rather than to the Greenstone copy.

Come formattare le liste

- **`Format lista-parte comandi`**
  - La prima parte (`list`) è obbligatoria ed identifica le liste alle quali applicare i comandi di formattazione
  - Search è la lista generata da una ricerca, mentre CL1, CL2, … sono le liste generate dal primo, secondo, … classificatore
  - La seconda parte (`parte`) è opzionale e specifica a quale parte della lista i comandi vanno applicati (HList, VList, DateList)
    - → **Ad es. `format CL4Vlist` si applica a tutte le VList in CL4**

Come formattare le liste

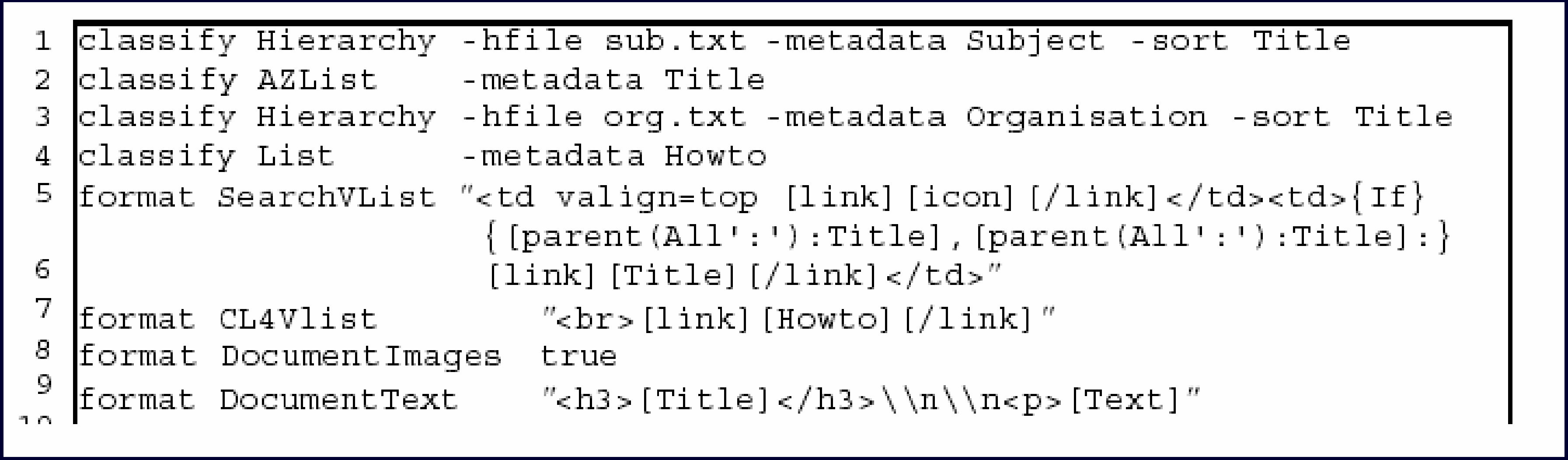
- `Comandi` è una stringa che specifica come formattare la lista
- Può contenere codice HTML, metadati ed i seguenti elementi



[Text]	The document's text
[link] ... *[/link]*	The HTML to link to the document itself
[icon]	An appropriate icon (e.g. the little text icon in a *Search Results* string)
[num]	The document number (useful for debugging).
[metadata-name]	The value of this metadata element for the document, e.g. *[Title]*

Un esempio [1/4]

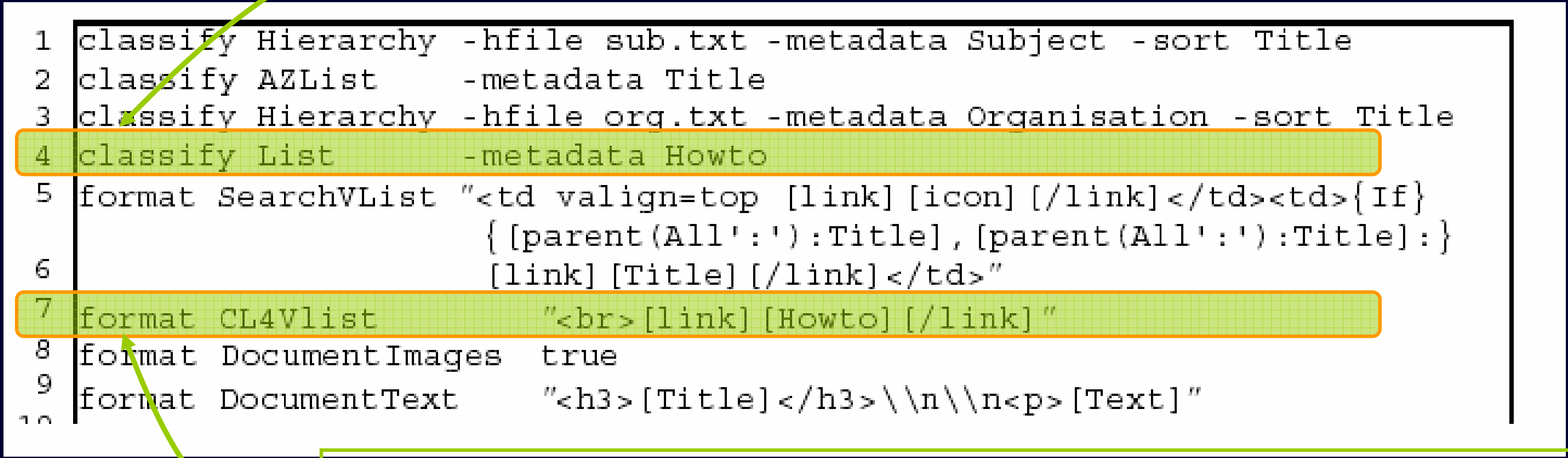
- **Esempio di classifiers e format commands della demo collection**



```
classify Hierarchy -hfile sub.txt -metadata Subject -sort Title
classify AZList    -metadata Title
classify Hierarchy -hfile org.txt -metadata Organisation -sort Title
classify List      -metadata Howto
format SearchVList "<td valign=top [link][icon][/link]</td><td>{If}
                    {[parent(All':'):Title],[parent(All':'):Title]:}
                    [link][Title][/link]</td>"
format CL4Vlist        "<br>[link][Howto][/link]"
format DocumentImages  true
format DocumentText    "<h3>[Title]</h3>\\n\\n<p>[Text]"
```

Un esempio [2/4]

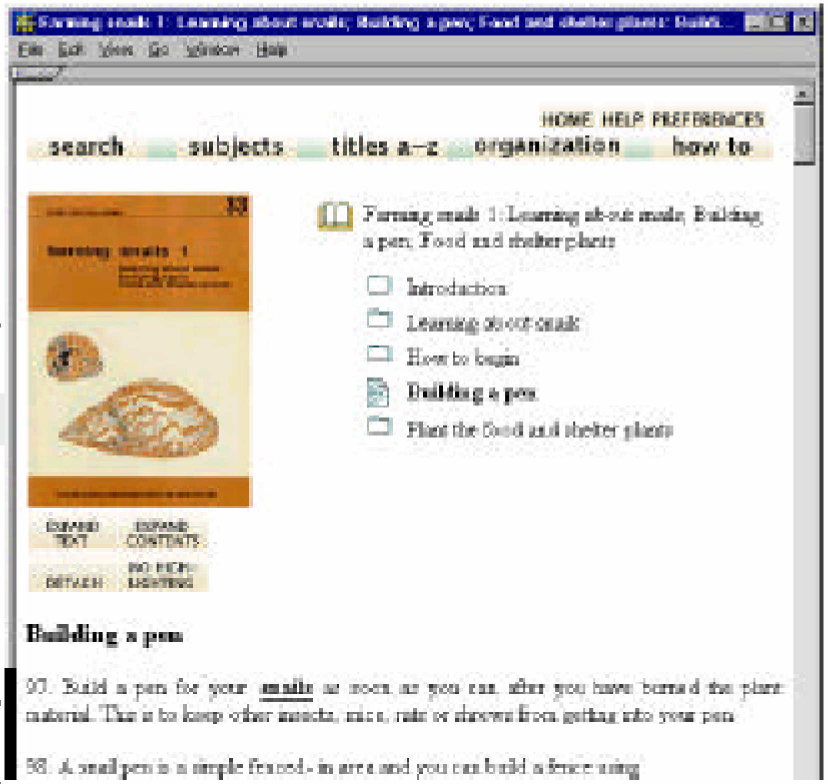
Howto classifier. È il quarto classifier (CL4)
È un List classifier che genera una lista di titoli di documenti



```
classify Hierarchy -hfile sub.txt -metadata Subject -sort Title
classify AZList    -metadata Title
classify Hierarchy -hfile org.txt -metadata Organisation -sort Title
classify List      -metadata Howto
format SearchVList "<td valign=top [link][icon][/link]</td><td>{If}
                    {[parent(All':'):Title],[parent(All':'):Title]:}
                    [link][Title][/link]</td>"
format CL4Vlist        "<br>[link][Howto][/link]"
format DocumentImages  true
format DocumentText    "<h3>[Title]</h3>\\n\\n<p>[Text]"
```

Comando di formattazione di CL4
Gli elementi figlio degli elementi top-level sono visualizzati come una VList
Ogni elemento si trova su una nuova linea e contiene il testo del campo Howto collegato al documento

Un esempio [3/4]



Farming snails 1: Learning about snails; Building a pen; Food and shelter plants: Build...

Farming snails 1: Learning about snails; Building a pen; Food and shelter plants

- Introduction
- Learning about snails
- How to begin
- **Building a pen**
- Plant the food and shelter plants

Building a pen

97. Build a pen for your snails as soon as you can, after you have burned the plant material. This is to keep other insects, mice, rats or shrews from getting into your pen.

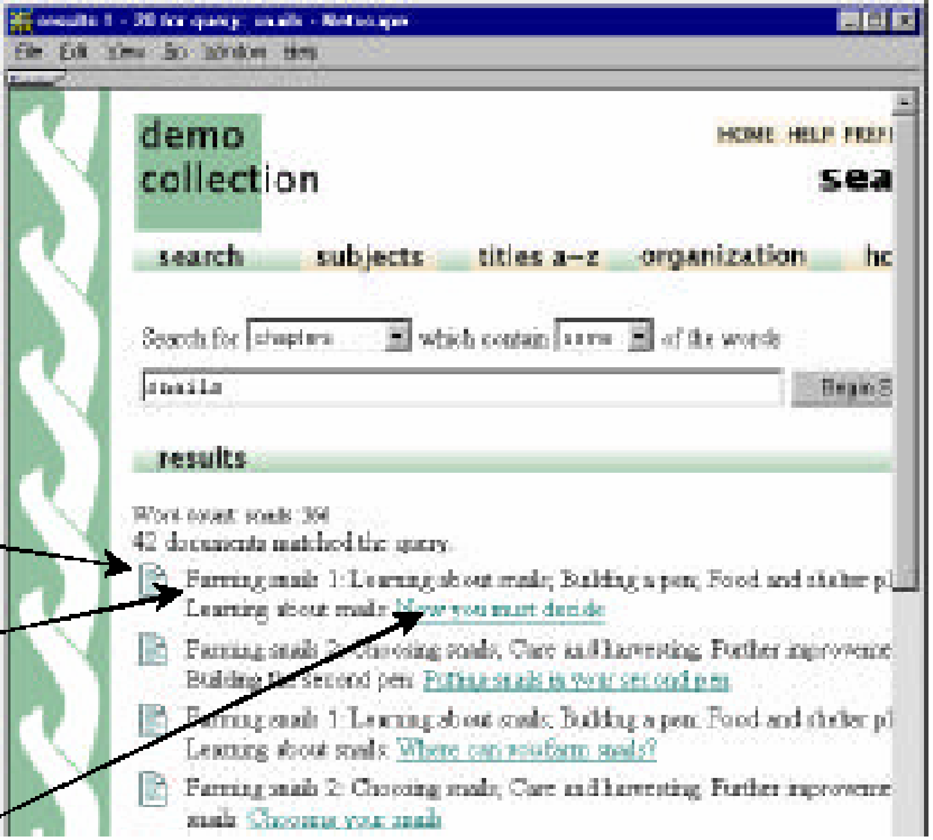
98. A snail pen is a simple fenced-in area and you can build a fence using

```
format DocumentImages true
```

```
format DocumentText
   "<h3>[Title]</h3>\\n\\n<p>[Text]"
```

```
classify Hierarchy
classify AZList
classify Hierarchy
classify List
format SearchVList "<td valign=top [link][icon][/link]</td><td>{If}
                    {[parent(All':'):Title],[parent(All':'):Title]:}
                    [link][Title][/link]</td>"
format CL4Vlist        "<br>[link][Howto][/link]"
format DocumentImages  true
format DocumentText    "<h3>[Title]</h3>\\n\\n<p>[Text]"
```

Un esempio [4/4]


File Edit View Go Window Help
demo collection
HOME HELP PREF
search subjects titles a-z organization
Begin Search
results

Search for chapters which contain some of the words

snails

Word count: snails: 394

42 documents matched the query.

Farming snails 1: Learning about snails; Building a pen; Food and shelter pl…
Learning about snails: How you must decide

Farming snails 2: Choosing snails; Care and harvesting; Further improvement…
Building the second pen: Putting snails in your second pen

Farming snails 1: Learning about snails; Building a pen; Food and shelter pl…
Learning about snails: Where can you farm snails?

Farming snails 2: Choosing snails; Care and harvesting; Further improvement…
snails: Choosing your snails

```
[link][icon][/link ]
```

```
[parent(All': ') :
Title]
```

```
[link][Title][/link]
```

```
classify Hie
classify AZI
classify Hie
classify List       -metadata Howto
format SearchVList "<td valign=top [link][icon][/link]</td><td>{If}
                     {[parent(All':'):Title],[parent(All':'):Title]:}
                     [link][Title][/link]</td>"
format CL4Vlist         "<br>[link][Howto][/link]"
format DocumentImages   true
format DocumentText     "<h3>[Title]</h3>\\n\\n<p>[Text]"
```