Using LinkController

Michael De La Rue

Published by ...

This manual is not yet complete, but it's better than nothing.

Manual Revision Code: $Revision: 1.22 $

Table of Contents

Introduction

LinkController is a system for checking links.

Most HTML pages contain references to other HTML pages (links). These allow the readers of those pages to locate other related resources (web pages etc.). Unfortunately the location of 'resources'[1] can change, resources can disappear completely or the system providing the resource can break. When this happens, the link which used to find them will no longer work. The only reliable way to detect this problem is to periodically check over the resources and take corrective action for the ones that have gone missing.

LinkController is designed to make that task much more efficient. It automates the task of checking which links have been broken for a period of time and then of finding which documents they occur in.

[1] resource is a general term for HTML pages and all of the other things that can be referenced by URLs

1 Getting Started

This section of the manual assumes that the programs that make up LinkController are already installed and working on your computer. If not, then See Section 7.1 [Setting up LinkController], page 23. We assume the standard setup where your system administrator runs the link checking for you. Other setups will need slightly different behaviour. Speak to the person who set up link controller.

The first thing to do is to run `configure-link-control` to configure the system. This will ask a series of questions about your configuration and create a configuration file which will be used by the various programs which make up LinkController.

Next you have to work out which links you are interested in. Do this by extracting the links from your web pages (see Section 4.1 [Extracting Links], page 13). The output file from this with the list of links found will be stored in the location you gave during configuration.

Assuming that you have the default install, your links will be automatically copied and checked over time following that.

After a short time (about a day) you will begin to get information about links which didn't work with `link-report --not-perfect`. After some more time (a week or so) you can use `link-report` to find out which links are really broken.

2 Configuration

2.1 Interactive Configuration

The `configure-link-control` program can be used by users to configure LinkController. This will ask you a series of questions and then generate a configuration file in your home directory. This is a good way to start configuring LinkController.

The configuration that is controlled by this program is related to reporting and fixing links. For other configuration see See Chapter 7 [Administration], page 23.

2.2 Setting Configuration Variables

If automatic configuration (see Section 2.1 [Interactive Configuration], page 5) doesn't work well enough for you, then you should manually change the configuration variables. All of the variable information is stored in the file '`.link-control.pl`' in your home directory or '`/etc/link-control.pl`' for system wide configuration. These locations are hardwired into the LinkController system and should only change if your administrator has done something strange. The configuration files are written directly in Perl (the programming language LinkController is written in). You can set the configuration variables by putting lines like this.

```
$::links='/var/lib/link_database.bdbm';
```

Please note the semi colon at the end of the line and the use of single quotes so that Perl doesn't do anything strange to your values.

2.3 LinkController Configuration Variables

This is a complete list of the configuration variables which a user should chnage in the '`.link-control.pl`' file. See Section 2.2 [Setting Configuration Variables], page 5, for how to do this.

`$::user_address`
: is the email address which the robot declares to the world as it goes around checking links. If you want to check links yourself, you must set this to a valid email address, because if something goes badly wrong, it is the only way for a user at another site to know how to contact you.

`$::base_dir`
: This is the base directory for all of the configuration files. If this variable is defined then the other variables will default as given below and do not need to be set individually.

`$::links` tells you what file is being used to store information about links. This could easily be a shared database used by everyone on your system. Defaults to '`$::base_dir/links.bdbm`'.

`$::schedule`
tells the system where to find the schedule file used to decide which links should be checked next and when that should be. You will need to set this and create the file in order to do link checking. Defaults to '`$::base_dir/schedule.bdbm`'.

`$::page_index`
This variable tells LinkController where to find the index which lists which links are contained on each page. Defaults to '`$::base_dir/page_has_link.cdb`'.

`$::link_index`
This variable tells LinkController where to find the index which lists which links are contained on each page. This is used during reporting to create the list of pages that should be repaired. It is also used during repair to decide which files have to be repaired. The file can be regenerated by running `extract-links`. Defaults to '`$::base_dir/link_on_page.cdb`'.

`$::infostrucs`
This variable points to the configuration file where definitions of infostructures are should be put Section 2.4 [Infostructure Configuration], page 6. Defaults to '`$::base_dir/infostrucs`'.

`$::link_stat_log`
This variable is the name of a file where important link status changes will be logged. The current definition is links which have just been discovered to be broken. This can be used in email notification Section 4.4 [Email Reporting], page 15. There is no default value for this file and it is not generated by default.

2.4 Configuring Infostructures

The infostructure configuration says where our web pages are stored and how they are accessed. It is kept in a separate file defined by the `$::infostrucs` configuration variable. The file is used by `extract-links` (see Section F.3 [Invoking extract-links], page 39) to find which files to get links from; it is used by `fix-link` (see Section F.4 [Invoking fix-link], page 40) to find out where files needing to be repaired are stored; it is used by `check-page` (see Section F.5 [Invoking check-page], page 41) to work out the base URI for any file being checked and finally it is used for certain reports in `link-report` (see Section F.1 [Invoking link-report], page 37).

The format of the file is one line for each infostructure with configuration directives separated by spaces. For example

```
directory http://example.com/manual /var/www/html/manual
www http://example.com/strange_database
```

The first directive describes how `extract-links` program should extract the links. It currently has three possible values. The value `www` means to actually use the given URL to download the web pages. The value `directory` means that `extract-links` should assume

that all of the files are stored in a directory and that the directory structure matches the structure of the infostructure. The final value `advanced` allows for further configuration at the cost of extra complexity. See Section 3.1 [Advanced Infostructure Configuration], page 9, for more information about this.

In the case where we use the `directory` directive, a third directive is present on each line with the full path to the base directory of the infostructure. In this case `fix-link` will be able to repair broken links in these files and `extract-links` will use direct file access to the file system when extracting links.

3 Advanced Configuration

There are various advanced ways to configure LinkController. These are mostly not needed for simple checking of a small collection of web pages. For larger sites and special situations however, they may well make life much easier.

3.1 Advanced Infostructure Configuration

Using more advanced configuration it is possible to skip over certain resources when we are doing link extraction and to ignore some of the links. You may want to skip over this section initially and come back to it only when you find that there are links or pages being checked that you would rather avoid.

For this section, we assume that you already know how to make basic Perl code. If not, then please read through the Perl manual pages '`perl`', '`perlsyn`' and '`perldata`'. You may find that the examples given below are sufficient to get you started.

In order to get `extract-links` to extract links using an advanced infostructure, you must use the `advanced` keyword. In the infostructure file. Infostructures not listed there will be ignored, but won't cause any harm.

Advanced configuration is in the '`.link-controller.pl`' configuration file by making definitions into the `%::infostrucs` hash. These look like the following

```
$::infostrucs{http://www.mypages.org/} = {
   mode => "directory";
   file_base => "/home/myself/www",
   prune_re => "^(/home/myself/www/statistics)" #ignore referrals
               . "|(cgi-bin)", #do CGIs separately
   resource_exclude_re => "\.secret$", #secrets shouldn't stay secret
   link_exclude_re => "^http://([a-z]+\.)+example\.com",
};

$::infostrucs{http://www.mypages.org/cgi-bin/} = {
   mode => "www";
   resource_exclude_re => "query", #query space is infinite!!
};
```

There are a number of keywordss that can be used.

'`mode`' This decides how to download the links. Either '`www`' or '`directory`'.

'`file_base`'
: If defined, this defines the directory which matches the URL where the infostructure is based. This must be defined if the mode is set to directory.

'`resource_include_re`'
: If defined, this regular expression must be matched by the *URL* for every resource before links will be extracted from it.

'`resource_exclude_re`'
: If defined, this regular expression must *not* be matched by the *URL* for every resource before links will be extracted from it.

'`link_include_re`'
: If defined, this regular expression must be matched by every *URL* found before it will be extracted and saved.

'`link_exclude_re`'
: If defined, this regular expression must *not* be matched by every *URL* found before it will be extracted and saved.

'`prune_re`'
: Used only in directory mode, this will completely exclude all files and sub-directories of directories matched by the regular expression.

N.B. the exclude and include regular expressions can be used together. For a match, the include regular expression must match and the exclude must not match. In other words excludes override includes.

In order for the infostructure to be used by `extract-links` an entry must still be made in the '`infostrucs`' file. For this use the `advanced` keyword. The second argument is a URL used to look up the definition in the $::infostrucs hash.

```
advanced   http://www.mypages.org/
advanced   http://www.mypages.org/cgi-bin/
```

The URL used here must match *exactly* the one used in the hash. It is important to note that '`directory`' and '`www`' definitions in the '`infostrucs`' file will override any advanced configuration given.

3.2 Authorisation Configuration

One problem when checking links, especially within an intranet situation is that some pages can be protected with basic authentication. In order to extract links from those pages or to simply know that they are there, we have to get through that authentication. By using the advanced Authorisation Configuration we can give LinkController authority to access these pages and allow link checking to work as normal.

> *Using this method to allow LinkController to work in an environment with authentication is inherently a security issue since authentication tokens must be stored, effectively in plaintext, in files. This risk may, however, not be much higher than the one that you currently accept, so this can be useful*

We can store the authentication tokens simply in the %::credentials hash which we can create in the '`.link-controller.pl`' configuration file. The keys in the hash are the exact realm string which will be sent by the web server. Each value of this hash is a hash with a pair of keys. The '`credentials`' key should be associated to the authentication token. The '`uri_re`' key should be a regular expression which matches the web pages you want to visit. For security reasons it shouldn't match any others.

```
$::credentials = {
  my_realm => { uri_re => "https://myhost.example.com",
                credential => "my_secret" }
} );
```

As a sanity check, every '`uri_re`' will be tried on '`http://3133t3hax0rs.rhere.com`' and '`http://3133t3hax0rs.rhere.com/secretstuff/www.goodplace.com/`'. If the expression matches then the credentials will be ignored. If you know enough to do this safely then you should definitely know how to get past this check. The owners of the domain '`3133t3hax0rs.rhere.com`' will just have to hack the code..

For more discussion about the security risks and how to mitigate them see the file '`authorisation.pod`' included with the LinkController distribution. If you didn't understand the security risk from the above description then probably you should consider avoiding using this mechanism.

3.3 Configuring CGI Programs

The CGI programs use the same configuration variables as the other programs, however, to avoid any confusion and related security problems, a perl script should be written which has the configuration variables hard wired in then runs the appropriate CGI program. `configure-link-cgi` is a program designed to set up such a script.

FIXME: this section needs to be rewritten.

4 Using LinkController to Check Links

This chapter covers in reasonable detail how to use each of the programs in LinkController.

4.1 Extracting Links

This section is written assuming that you are using a standard HTML infostructure in a directory or on the World Wide Web

The first part of using link controller is to extract the links. When doing this, a pair of index files is built which list which URLs happen on which pages along with a file listing all of the URLs in the infostructure.

FIXME: compare and contrast multi-user configuration with single user

The first stage of the process is done by `extract-links`[1].

There are two modes for extract links `directory` and `www`. The key difference between them is that the latter actually downloads from a server so it is less efficient but will work in more circumstances and is more likely to represent your site as seen by users. This is assuming that all of your WWW pages are interconnected so it can find them.

FIXME : need to describe modes of operation of extract link

`extract-links` creates three files. The first two files ('`*.cdb`') are the index files for your infostructure and are located wherever you have configured them to by default they are called '`link_on_page.cdb`', '`page_has_link.cdb`'. The third file is the database file '`links.db`'. `extract-links` can also optionally create a text file which lists all of the URLs in the infostructure, one per line.

There are a number of other ways of using `extract-links` and it has many options. See Section F.3 [Invoking extract-links], page 39, for more information about using extract links.

4.2 Testing Links

If you are using someone else's link information then you may be able to skip this part and go straight on to the next one on generating reports. If not then the next stage is to test your links using `test-link`.

Testing links takes a long time. Reporting of broken links will not begin until after several days. This is a deliberate feature of LinkController. Most problems that will be found in a well maintained web page will be temporary configuration or system problems. By wainting

[1] the command names in link controller are quite long.. you might want to make your life easier by using command completion which will finish what you have started.. once you've typed a little of the command use *`escape escape`* or *`tab`* depending on your shell.. if this doesn't work then you may like to upgrade to a newer shell such as bash or zsh.

to report problems we give people responsible for the other end of the problem link a chance to repair their resources. Once we have made this decision, we may as well check slowly and in a way which will reduce the amount of network bandwidth LinkController uses at a given time and so its impact on other people's Internet usage.

The key program which you want to use is `test-link`. I run this from a shell script which directs its output to a log file

FIXME actually I now just use a cron job.

```
#!/bin/sh
#this is just a little sample script of how I run the program.

LOGDIR=$HOME/log
test-link >> \
        $LOGDIR/runlog-`/bin/date +%Y-%m-%d`.log 2>&1
#assumes the use of a gnu style date command which can print
#out full dates.
```

And I run this shell script from my '`crontab`' with a command like this

```
42 02 * * *     /..directories./run-daily-test.sh
```

The string `/..directories./` should be replaced with the directory where you have the script. Remember to make the script executable.

This will now run until completion each night. However, you should make sure that it does actually finish. If you have too many links to check in the given time, then you can end up with a backlog and the system will take a long time to stop. To avoid this, either make testing less frequent or make checking run faster. This will have to be done by editing the program itself at present.

The `test-link` program has a number of options. These control the limits on checking and the speed of checking. See Section F.3 [Invoking extract-links], page 39, for more information on these.

4.3 Reporting Problems

The easiest way to find out which links are broken is to use the command line interface. The simplest report you can generate is just a list of all the known broken links. Do this like so:

```
link-report
```

On the system I'm testing on right now, this gives:

```
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/cgi
        http://www.ippt.gov.pl/docs-1.4/cgi/examples.html
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/curr
ent/httpd_1.4_irix5.2.Z
        http://www.ippt.gov.pl/docs-1.4/setup/PreExec.html
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/curr
ent/httpd_1.4_linux.Z
        http://www.ippt.gov.pl/docs-1.4/setup/PreExec.html
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/curr
```

```
ent/httpd_1.4_osf3.0.Z
        http://www.ippt.gov.pl/docs-1.4/setup/PreExec.html
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/curr
ent/httpd_1.4_solaris2.4.Z
        http://www.ippt.gov.pl/docs-1.4/setup/PreExec.html
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/curr
ent/httpd_1.4_solaris2.4.tar.Z
        http://www.ippt.gov.pl/docs-1.4/setup/PreCompiled.html
Sorry, couldn't find info for url file://ftp.ncsa.uiuc.edu/Web/httpd/U
nix/ncsa_httpd/current/httpd_1.4_source.tar.Z
please remember to check you have put it in full format
broken:-       file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/docu
ments/usage.ps
        http://www.ippt.gov.pl/docs-1.4/postscript-docs/Overview.html
..etc...
```

Which just tells you which links are broken. We also know which page they are broken on and can go and look at that on the World Wide Web or directly as a file on the server.

There are many different options which control the output of `link-report`. These include options which select which kinds of problems to report, options which select which pages to report from and options which allow other output formats such as HTML. See Section F.1 [Invoking link-report], page 37, for more information about these.

For more advanced reporting and editing of documents with broken links you may want to use the Emacs interface (see Chapter 6 [Emacs], page 21).

4.4 Email Reporting of Newly Broken Links

It's possible to arrange automatic reporting by email of links which have become newly broken. This is done by getting `test-link` to make a list of links that become broken using the '`$::link_stat_log`' variable (see Section 2.3 [Configuration Variables], page 5) and calling `link-report` to report on those links.

Typically, you may don't want to have a report every time that `test-link` runs, but probably once a day instead. In this case, run a script like the following from your crontab.

```
#!/bin/sh
STAT_LOG=$HOME/link-data/stat-log
WORK=$STAT_LOG.work
EMAIL=me@example.com
mv $STAT_LOG  $WORK
if [ -s $WORK ]
then
   link-report --broken --url-file=$STAT_LOG |
      mail -s "link-report for `date`" $EMAIL
fi
```

Every time that this script is run, it will rename the status change log file and then mail a report with all of the new broken links to the specified email address.

4.5 Examining Individual Files

When you have just written an HTML page, you often want to check it before you put it up for use. You can do this immediately using the `check-page` program. Simply run something like

```
check-page filename.html
```

And it will list all of the links that it is unsure about along with the line number the problem occurred on. This program works particularly well when you editing with Emacs (see Section 6.2 [check-page in Emacs], page 21).

4.6 Repairing Links

The program responsible for repairing links is `fix-link`. It simply accepts two URLs and changes all of the occurrences of the first link in your documents into the second link. It assumes that you have permission to edit all of the problem files and that there is a replacement link. For example

```
fix-link http://www.ed.ac.uk/~mikedlr/climbing/ \
        http://www.tardis.ed.ac.uk/~mikedlr/climbing/
```

Typed at the shell prompt would have updated the location of my Climbing pages when they moved some while ago and

```
fix-link http://www.tardis.ed.ac.uk/~mikedlr/climbing/ \
        http://scotclmb.org.uk/
fix-link http://www.tardis.ed.ac.uk/climb/ \
        http://scotclmb.org.uk/
```

Will change them to the very latest location. More information about `fix-link` can be found in See Section F.4 [Invoking fix-link], page 40.

At present, there's no facility for automatically updating the databases when you do this. Instead, you have to run `extract-links` after some time so that new links are noticed. In practice this doesn't matter because you shouldn't be creating new pages with broken links and can check that you don't with `check-page`. A later version of LinkController will may change this.

The other way to fix links is to edit the files by hand. This is the only solution where a link has disappeared forever and so text changes have to be made to the web site. This can be made more convenient by using the '`link-report-dired`' emacs module included in the distribution. This is covered elsewhere in this manual (see Chapter 6 [Emacs], page 21).

4.7 Making Suggestions

A link in the database can have suggestions associated with it. These are normally alternative URLs which somebody or something has decided would make a good replacement for the URL of the Link. Humans can add to the database with the `suggest` program. For example use:

```
suggest file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/htt
pd_1.4_linux.Z \
        http://delete.me.org/
Link suggestion accepted.  Thank you
```

If you try the same thing again you get

```
suggest file://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/htt
pd_1.4_linux.Z \
        http://delete.me.org/
Already knew about that suggestion.  Thanks though.
```

These suggestions will make it easier for others to repair links, especially if they are using the CGI interface.

4.8 CGI Interface

The CGI interface is not fully developed and has a number of issues related to security to be considered. I have however used it and shown that it can work, so if you want to you could try the same. The two programs `fix-link.cgi` and `link-report.cgi` replace the normal ones `fix-link` and `link-report`. They should be interfaced through an HTML page which feeds the needed information to `link-report.cgi`.

The main security question is how to do authentication of the user. This will have to be set up using the features of the web server. You should not leave these programs available for non-authenticated users since that would give them the ability to edit your web pages directly and probably do worse.

5 Interfacing to other programs

Probably not all of your links are directly in web pages. If this is the case, it's still possible to use LinkController to check those links, but it won't be possible to use the repair facilities.

In this case, you have to generate the list of URIs you want checked yourself. This should be a file with one URI per line. Then `extract-links` can be used to import those links into LinkControllers database. For example, if you had put those links into the file '`links`' the following command would import them.

```
extract-links --in-url-list=links
```

Now, when you want to report on your links you can give the links file as an argument to link-report and it will only report those links which are in your file. This can be done with the following command

```
link-report --url-file=links
```

The usual options can be given to control which links are reported for example '`--all-links`' to list all links (see Section F.1 [Invoking link-report], page 37).

Another possibility for interfacing to programs is to use output from LinkController to automatically remove links from your web pages. That would be a very suitable solution, for example, if you keep a list of links to other related pages, but don't mind if some of them disappear temporarily.

In this case, it's probably best to use the `link-report` option for machine oriented output '`--uri-report`' and to choose either the '`--broken`' report for deleting links or the '`--good`'[1] option to choose which links should be shown on your web pages. For example, run something like the following each night from your '`.crontab`' file.

```
link-report --url-file=links --uri-report --broken \
   | automatic-link-deleter
```

You should probably mail someone with the information that the link has been deleted so that if there's an easy way to fix it they can do that.

[1] Note: this option doesn't output links which can't be checked.

6 The Emacs Interface

LinkController's reporting system is designed to be independent of the interface to it, and often the shell interface will be all that is needed. However another convenient interface is through `emacs`. There are two parts to this integration.

6.1 Finding Files with Broken Links

There is a special Emacs mode called `link-report-dired` written for locating files with broken links. The mode is based on `find-dired` and works very similarly. It runs the program `link-report` with an option which makes it list file names in the same way as the `ls` program does. The user can then move around the buffer as normal in Emacs and enter files using a single key press (normally *`f`*).

6.2 Finding Broken Links in Files Within Emacs

The program `check-page` was specially designed so that it outputs in a format which can be read by Emacs' `compile` mode. You can use it within Emacs and then step from error to error correcting them.

To do this, after you have set up your system and run '`test-link`' a few times. checking use the command *`M-x compile RET check-page filename RET`* . You will now see another buffer open up with all of the errors shown there. You can use the key *`M-‘`* (that's a real back quote, not an apostrophe) to step between errors.

The one problem with '`check-page`' is that if you have just created a file containing new links it should really verify them by testing each one. This makes it more suitable for use during link correction of existing pages than during writing new pages.

7 Administration

There are various aspects of administration. This is mostly related to testing links.

7.1 Setting up LinkController

This chapter is aimed at administrators setting up LinkController or who want to have a better understanding of the way that their installation is set up.

The first stage is to actually build and install the programs. This is covered in the document '`INSTALL`' which is included with the distribution.

Once you have installed the software, the next step is to configure LinkController so that it knows where you have all of your data. The program `default-install` provides one model of this.

7.2 Default Installation

Running `default-install -all` should set everything up correctly. There are various variations on this command which do different things, but the summary is

- Create a linkcont user-id and group which will be used for running programs
- Create a working directory where LinkController will keep it's data
- Create configuration files, especially /etc/link-control.pl
- Create cron scripts which will run LinkController automatically.

Using this command it is also possible to activate users and groups e.g. `default-install -user` *username* or `default-install -group` *groupname* in which case the specified users will become a member of the `linkcont` group.

7.3 User Administration

User administration is really only needed if you are running link testing centrally for your users. This makes sense since it means that if several users have a link to the same place (likely in any given site) then you will only have to check that link once.

In this case, the important question is which links are copied into the checking database. This is controlled by the program `copy-links-from-users` and decides copies data from users which are in the `lnctusr` group.

The command `default-install` can be used to manipulate which users are in the group e.g. `default-install --user` *username* or `default-install -group` *groupname* in which case the specified users will become a member of the lcntusr group.

Another form of user administration is limitation on which users have access to the database. This can be done with normal file permissions. There isn't any specific control to stop users from seeing which links other users have put into the database.

7.4 Cron Scripts

In order to be effective, link testing should be done every day. Furthermore, it is a good idea to do the testing at low usage times, which normally means at night. For this reason normally a cron script will be used.

- copy the links from each user with `copy-links-from-users`
- add them to the database with `extract-links --in-url-list`
- build a schedule for testing the links with `build-schedule`
- test the links with `test-link`

The program `default-install` can create these scripts Section 7.2 [Default Installation], page 23.

7.5 Link Database Maintenance

For the most part the link database shouldn't need much maintenance. There are a number of cases where it might, however. If it becomes corrupt, you may try the db_recover command. Probably, however, it's just better to recover the database and link checking schedule from a recent backup. Time is not really critical since the work is normally easy to regenerate. You should make sure that link checking doesn't run in at the time you do backups, however.

The other thing is that occasionally you may want to recover space in the link database by dumping and un-dumping it. See the Berkeley database documentation for more details.

7.6 Link Ageing

Sometimes we have a link which is no longer in use within our infostructure. However, it's not a good idea to throw away information about it immediately. It could be that certain files have been temporarily deleted and will come back. Alternatively, a link could have been found broken and been corrected, but someone has a copy of the old page. When they re-install the copy, we will have to deal with that link again.

If we had not kept that link around and they immediately do a link check on their document they may see nothing wrong, and, because of the nature of LinkController, we won't start reporting the link as a problem until some days later, when we have confirmed that it really is broken.

If, on the other hand, we keep checking all of the links which have ever been in our web pages, we will cause ourselves considerable extra work.

To handle this, links are aged. Once a link has reached greater than a certain age, `test-link` will not be checked it any more. Once the link has reached a much larger age, it will be completely deleted from the link database. The age is reset to nothing each time the link is extracted from the infostructure, so links which are still in use will continue to be checked.

In the meantime, the link will still be repeatedly scheduled based to it's normal checking time. This causes us to examine it quite regularly, but that is okay, since we won't actually check it.

By default links which have not been refreshed will be ignored after one week and will be deleted from the database after two months.

Appendix A Robots and Sensible Behaviour

The most important thing about a program like this is to realise that if you set it up incorrectly and used it in the wrong way, you could upset a large number of people who have set up their web servers in the assumption that they would be used normally by human beings browsing through on Netscape.

It is true that LinkController is very careful to limit resource usage on remote sites, but the other site may not know that or may have a real reason not to want their pages visited too often.

Probably it's true that the only safe way forward is for every WWW site to begin to set up robot defences and detect when someone starts to download from them at an unreasonable rate and then cut off the person doing the downloading. I suggest that you don't make people have to do this to protect themselves against you for at least two reasons.

- respect for the person's time
- a wish not to be the person who is cut off

There are probably many other reasons, but that's one for the good side in you and one for the selfish. What more do you need.

For suggestions about what constitutes 'correct' behaviour, it's worth seeing the Robots World Wide Web page. `http://www.robotstxt.org/wc/robots.html`

There are a number of points which make LinkController relatively safe as a link. These are all related to the design and limitations on `test-link`.

- `test-link` does *not* recurs. It only tests links that are specifically listed in the schedule database.
- There is a limit to the number of links that will be tested in one run. This defaults to 1000, but can be configured.
- The schedule for link testing is designed to spread the testing of links across time
- The testing system will not test links at a given site faster than a certain rate.

The last limitation is inherited from the `LWP::RobotUA` module and the documentation for that covers the details of how it works. `test-link` tries to re-order testing of links as needed so that a limit on the rate of visits to one site does not cause a limit on overall testing speed.

Appendix B Uncheckable Links

Some links can't be checked because the target url doesn't have an easy method of verification or because the link checker doesn't have the facilities needed for verification. Examples of this include '`mailto`' and '`news`' URLs.

Although it's possible to verify, to a certain degree, many mail addresses the only absolute way to check that a mail address reaches the person it's meant to reach is to send a mail and ask for them to reply. Obviously, if everybody checking every link in the world started to do this some unlucky recipients would get very upset at being bombarded with so much mail.

A low level of verification could be done in some circumstances. This requires that the persons actual mail server (the place where their mail is kept) can be contacted directly and is willing to be helpful. This will be implemented in a later version of LinkController.

Checking news urls (not possible at the present moment anyway) requires access to a *correctly set up* news server which has the feed for that newsgroup. Even then, it doesn't talk about the access for the end user.

Other links cannot be checked because `libwww-perl` doesn't yet support them. In this case the solution is to add support to `libwww-perl`.

Appendix C Absolute and Relative URIs

LinkController is designed to handle absolute and relative URIs in a consistent but sensible fasion, but unfortunately there isn't any totally clear correct way. For link extraction we simply conver relative URLs to absolute form. For link testing, this means that we don't ever think of relative URLs. For link fixing on the other hand the situation is more complex. For this reason there is a `--relative` option to `fix-link`.

If we run `fix-link` without the `--relative` option then we only substitute absolute links in the existing document to the link given on the command line. This is safer because the subsitution is unlikely to mistake other strings which accidentally match the link.

If we run `fix-link` with the `--relative` option on the other hand then we will handle relative links. What this means depends on whether the links to be fixed can be expressed as links relative to the pages being fixed.

If the original (broken) link can be expressed as a relative link then we will do substitutions where we find relative links. If the target (corrected) link can be expressed as a relative link then we will always substitute broken links with a relative link.

Taken together, this means that if a resource has moved to the same server as your pages, substitution with the `--relative` option will correction convert all of your absolute links into relative links and if a resource has moved from your server to another then we will correctly substitute relative links with absolute links.

The only undesirable effect would be if a resource is moved within your pages and you have a mixture of relative and absolute links to that resource (e.g. for absolute links on page which is mirrored on other sites). In this case, first do substitution without the `--relative` option and then afterwards with the `--relative` option.

Appendix D Bugs and bug reporting

This version of LinkController is still in early development. There are many changes to come. Undoubtedly there are many bugs in the software already and will soon be more.

A bug is when

- the software doesn't do something the documentation says it should
- the software does something the documentation says it shouldn't
- the software does something surprising and that isn't documented
- the software does something strange but the documentation doesn't explain why
- it is difficult or impossible to understand what the documentation is trying to say

some of these mean fixing the documentation and some the software. All of them are bugs and should be reported and fixed.

If you find a bug, I will be grateful to hear about it. Even if you don't know how to fix it or anything, it is useful to know what is wrong so that other people don't get caught out but *read the BUGS file first* please. If the bug is listed there then the only useful thing that you can do is fix it. If you do this and contribute it to me then that is very useful.

When you report a bug, please tell me what release of link controller you were using. This is the number which was in the name of the file that LinkController came in. If your problem was with a specific program, please also run '`program --version`' and send the output. This tells me exactly which version of that program you were running.

Since this is a developers release, I'd hope most users would be able to make some level of fixes. If you do this, send me context differences (use '`diff -u`' if it works or try '`diff -c`' otherwise). I use CVS, so as long as I know which version you have I will be able to find the original file and see your changes. However it's also important to explain them because I won't be able to use them unless I (relatively stupid computer type) understand them.

Send bug reports to the address you get by changing words into punctuation in the following.

```
link minus controller at scotclimb dot org dot uk
```

This mailing address is sent only to me right now, but may become a list in future. Use my (Michael De La Rue) personal address to contact me please. N.B. I am **extremely** inefficient about answering email. Don't worry if you don't get a reply.

The ideas, and history

Appendix E History

LinkController was originally inspired by MOMspider and having the MOMspider code available was very useful when starting the creation of this kit, but, it shares almost no code with MOMspider, other than what has comes to it from the LibWWW-Perl library.

Philosophically, the MOMspider heritage is obvious in the wish to handle big jobs efficiently. In the working practice there are far more differences than similarities, partly caused by Perl language changes.

I decided to completely separate the exploration of the local infostructure, looking for links to be checked, from the actual checking process. This means that checking can be spread over a large number of days and still run efficiently.

The basic aim of this link checking kit is to be able to efficiently handle any size of link checking job. At the bottom end we have checking new pages as they are written. Here we want to use information from previous checks to avoid having to check all of each page every time. At the other end we have massive info structures (sites) which deal in many thousands of links and could not possibly all be checked in one day. For this latter case the aim is to be able to efficiently spread the link checking load into all available low usage periods.

My primary aim in writing this was not to write very efficient code for the small scale case (takes minimum time to do everything), but rather code which would scale well. If your system can check 1000 links in two days, it will hopefully be able to check almost 7000 links in two weeks. I'm trying to make sure all data structures which grow with the number of links are kept on disk.

E.1 Acknowledgements

Although I wrote this system by myself, this would not have been nearly as easy and almost certainly wouldn't have ever been finished without the help of the following people and organisations.

Esoterica Internet Portugal

Esoterica provided me with full access to the Internet in Portugal and use of their computers for free which allowed me to keep up on both this software and the Linux Access HOWTO. In particular I'd like to thank all of the members of staff who helped me very much. These people include Mario Francisco Valente (the instigator of Mini Linux) who first agreed to me using their kit, set me up to use their machines, and along with Luis Sequeira provided a sounding board for some ideas. Luis also provided the odd lift home in the evening. Also Martim de Magalhaes Pereira and Mr Mendes. See them all on

http://www.esoterica.pt/esoterica/quemsomos.html

For more about esoterica (Internet Services in Portugal) see:

http://www.esoterica.pt/esoterica/

These pages are in Portugese[1] of course.

IPPT PAN Poland

Thanks go to IPPT PAN (part of PAN - Polska Akademia Naukowa) in Poland and in particular Piotr Pogorzelski who allowed me use of facilities for testing this software, provided a willing victim for having his web pages tested and made a number of suggestions which have been incorporated into the software.

The Tardis Project

Supported by the Computing Science department of the University of Edinburgh, the Tardis project provides an experimental framework in which students, former students and other related people to do their own work on fully Internet connected Unix and Linux hosts.

The use of the facilities of the Tardis Project has made it much easier for me to develop software like this. In particular, the large amount of disk space the administrators have allow me to use is very useful.

Other Free Software Authors

It is through the software provided by the Free Software Foundation (such as the `gcc` C compiler, Emacs, the file utilities), the authors of the various packages which make up a working Linux System (Linux by Linus Torvalds, Alan Cox, etc.... filesystems and support by Theodore Tytso, Stefan Tweedie etc.. Linux-Libc by HJ Lu, based on GNU `glibc` from the FSF.. the list is indefinite) and the authors of Perl and its modules, especially Gisle Aas and Martijn Kostler for LibWWW-Perl that I was able to set this up.

I'd particularly like to thank Tim Goodwin the author of the Perl CDB module who made and accepted a number of alterations to that, at my request. These alterations made this package simpler to write and easier to maintain.

The Free Software Foundation web pages are at

http://www.gnu.ai.mit.edu/

[1] Whilst the above names are mangled here. See the correct versions in the original texinfo or on the Web pages.

Appendix F Invoking the LinkController Programs

Because they use the Perl `Getopt::Mixed` module, all of the LinkController command line programs respond to the standard POSIX style command line options. At least the following two options will be implemented.

'`--help`' This option will give a list of all of the options understood by the program along with brief explanations of what they do.

'`--version`'
This option will give some version information for the program.

You can use the '`--help`' option to get help on each program, for example:

```
extract-links --help
```

You can then use that information to get the program to do what you want.

F.1 Invoking link-report

The '`link-report`' program prints out status information about links allowing the user to see what needs to be fixed. The default is to print out all of the broken and redirected links that currently occur on the users web pages and which are either redirected or broken.

Before running '`link-report`' you should probably use `test-link` (see Section F.2 [Invoking test-link], page 38) to check which links are broken. That may not be needed if your system administrator does it for you. After you have identified broken links you may want to use `fix-link` (see Section F.4 [Invoking fix-link], page 40) to repair the broken links.

The primary configuration file used by `link-report` is the '`.link-control.pl`' file. This tells it where the schedule file and LinkController database are. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file.

In the case of the '`--long-list`' report, a second configuration file, the '`infostrucs`' file, is used. This contains the information needed to know where to extract links from by default. See Section 2.4 [Infostructure Configuration], page 6, for more details on configuring this.

FIXME this section should give a better description of each option.

```
link-report [options]

 -V --version            Give version information for this program
 -h --help --usage       Describe usage of this program.
    --help-opt=OPTION    Give help information for a given option
 -v --verbose[=VERBOSITY] Give information about what the program is
                          doing.  Set value to control what information
                          is given.

 -U --uri=URIs           Give URIs which are to be reported on.
 -f --uri-file=FILENAME  Read all URIs in a file (one URI per line).
 -E --uri-exclude=EXCLUDE RE Add a regular expressions for URIs to
```

```
                        ignore.
-I --uri-include=INCLUDE RE Give regular expression for URIs to check
                        (if this option is given others aren't
                        checked).
-e --page-exclude=EXCLUDE RE Add a regular expressions for pages to
                        ignore.
-i --page-include=INCLUDE RE Give regular expression for URIs to check
                        (if this option is given others aren't
                        checked).

-a --all-links          Report information about every URI.
-b --broken             Report links which are considered broken.
-n --not-perfect        Report any URI which wasn't okay at last test.
-r --redirected         Report links which are redirected.
-o --okay               Report links which have been tested okay.
-d --disallowed         Report links for which testing isn't allowed.
-u --unsupported        Report links which we don't know how to test.
-m --ignore-missing     Don't complain about links which aren't in the
                        database.
-g --good               Report links which are probably worth listing.

-N --no-pages           Report without page list.
   --config-file=FILENAME Load in an additional configuration file
   --link-index=FILENAME Use the given file as the index of which file
                        has what link.
   --link-database=FILENAME Use the given file as the dbm containing
                        links.

-l --long-list          Where possible, identify the file and long
                        list it (implies infostructure).  This is used
                        for emacs link-report-dired.
-R --uri-report         Print URIs on separate lines for each link.
-H --html               Report status of links in html format.
```

F.2 Invoking test-link

The `test-link` program tests all of the links in the LinkController database storing information about any problems found. It works as a robot contacting the servers where the target of each link is stored and verifying that the resource the link points to is really there.

Before running `test-link` you should probably use `extract-links` (see Section F.3 [Invoking extract-links], page 39) to collect all of the links you want to test and then `build-schedule` (see Section F.6 [Invoking build-schedule], page 42).

The configuration file used by `test-link` is the '`.link-control.pl`' file. This tells it where the schedule file and LinkController database are. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file.

FIXME this section should give a better description of each option.

```
test-link [arguments]

 -V --version            Give version information for this program
 -h --help --usage       Describe usage of this program.
    --help-opt=OPTION    Give help information for a given option
 -v --verbose[=VERBOSITY] Give information about what the program is
                          doing.  Set value to control what information
                          is given.
 --quite -q --silent     Program should generate no output except in
                          case of error.
    --no-warn            Avoid issuing warnings about non-fatal
                          problems.

 -c --config-file=FILENAME Load in an additional configuration file
 -u --user-address=STRING Email address for user running link testing.
 -H --halt-time=MINUTES  stop after given number of minutes

    --never-stop         keep running without stopping
    --no-robot           Don't follow robot rules.  Dangerous!!!
 -w --no-waitre=NETLOC-REGEX Home HOST regex: no robot rules..
                          (danger?)!!!
    --test-now           Test links now not when scheduled (testing
                          only)
    --untested           Test all links which have not been tested.
    --sequential         Put links into schedule in order tested (for
                          testing)
 -H --halt-time=MINUTES  stop after given number of minutes
 -m --max-links=INTEGER  Maximum number of links to test (-1=no limit)
```

Several of the options could potentially lead to overloading networks and even other people's computer systems:

Don't use –no-robot, except for when you are doing local testing (that is, you aren't connected to the internet proper).

Don't use –never-stop or –test-now except when you are watching what is happening.

Generally you should be somewhat careful about running this program since it does automatically connect to other servers on the internet. Reasonable care has been taken to ensure it does this in a responsible way, but you must make sure that anybody who is inconvenienced has a good route for communicating this problem back to you.

F.3 Invoking extract-links

The `extract-links` program walks through the users web pages collecting all of the links from those pages and storing them into a database for later checking by the `test-link` program (see Section F.2 [Invoking test-link], page 38). It can also list the links found into a given file.

After running `extract-links` you should use `build-schedule` (see Section F.6 [Invoking build-schedule], page 42) which will make sure that any new links discovered are scheduled for checking..

There are two configuration files used by `extract-links`. The '`.link-control.pl`' file is the first. This tells it where the various files it uses are. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file. The second file is the '`infostrucs`' file. This contains the information needed to know where to extract links from by default. See Section 2.4 [Infostructure Configuration], page 6, for more details on configuring this.

FIXME this section should give a better description of each option.

```
extract-links [arguments] [url-base [file-base]]

 -V --version            Give version information for this program
 -h --help --usage       Describe usage of this program.
    --help-opt=OPTION    Give help information for a given option
 -v --verbose[=VERBOSITY] Give information about what the program is
                          doing.  Set value to control what information
                          is given.
 --quiet -q --silent     Program should generate no output except in
                          case of error.

 -e --exclude-regex=REGEX Exclude expression for excluding files.
 -p --prune-regex=REGEX  Regular expression for excluding entire
                          directories.
 -d --default-infostrucs handle all default infostrucs (as well as ones
                          listed on command line)

 -l --link-database=FILENAME Database to create link records into.
 -c --config-file=FILENAME Load in an additional configuration file

 -o --out-url-list=FILENAME File to output the URL of each link found to
 -i --in-url-list=FILENAME File to input URLs from to create links
```

F.4 Invoking fix-link

The `fix-link` program is designed to repair a broken links across all of the files which LinkController is managing. It does this by looking up index files and seeing files contain the broken link then doing a textual substitution in each of these files. This makes it much faster than searching through all of the files in a set of web pages to see which pages have the broken link.

In order to work properly, `extract-links` (see Section F.3 [Invoking extract-links], page 39) must have been run first to build up the index databases used by `fix-link`.

There are two configuration files used by `fix-link`. The file '`.link-control.pl`' is the first. This tells it where the other configuration file and index files are. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file. The

second file is the '`infostrucs`' file. This contains the information needed to relate broken links to the files which need to be repaired. See Section 2.4 [Infostructure Configuration], page 6, for more details on confiuguring this.

```
fix-link [options] old-link new-link

 -V --version            Give version information for this program
 -h --help --usage       Describe usage of this program.
    --help-opt=OPTION    Give help information for a given option
 -v --verbose[=VERBOSITY] Give information about what the program is
                          doing. Set value to control what information is
                          given.
 -q --quiet --silent     Program should generate no output except in
                          case of error.
    --no-warn            Avoid issuing warnings about non-fatal
                          problems.

    --directory=DIRNAME  correct all files in the given directory.

 -r --relative           Fix relative links (expensive??).
 -t --tree               Fix the link and any others based on it.
 -b --base=FILENAME      Base URI of the document or directory to be
                          fixed.

    --config-file=FILENAME Load in an additional configuration file
```

F.5 Invoking check-page

Check page is useful where broken links in files need to be manually corrected. It outputs a list of line numbers where interesting links occur allowing the user to find those lines and correct the mistakes. The output format is compatible with the `emacs compile` mode which allows fast access to the problem locations.

There are two configuration files used by `extract-links`. The file '`.link-control.pl`' is the first. This tells it where the link database is. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file. The second file is the '`infostrucs`' file. This allows `check-page` to know what the base URI of the file being checked is and so check relative links within the page corectly. See Section 2.4 [Infostructure Configuration], page 6, for more details on configuring this.

```
check-page [options] filename...

 -V --version            Give version information for this program
 -h --help --usage       Describe usage of this program.
    --help-opt=OPTION    Give help information for a given option
 -v --verbose[=VERBOSITY] Give information about what the program is
                          doing.  Set value to control what information
                          is given.
```

```
-r --redirect           Report links which are redirected.
-m --ignore-missing     Don't complain about links which aren't in
                        database.

   --link-index=FILENAME Use the given file as the index of which
                        file has what link.
   --link-database=FILENAME Use the given file as the dbm containing
                        links.
```

F.6 Invoking build-schedule

The `build-schedule` program makes a schedule for testing links. If run with no options it will make sure that all the links in the LinkController database will be checked at some point in the future.

Before running `build-schedule` you should probably use `extract-links` (see Section F.3 [Invoking extract-links], page 39) to collect all of the links you want to test. Afterwards you should use `test-link` to check which ones are broken (see Section F.2 [Invoking test-link], page 38).

The configuration file used by `build-schedule` is the '`.link-control.pl`' file. This tells it where the schedule file and LinkController database are. See Section 2.2 [Setting Configuration Variables], page 5, for how to control the contents of this file.

```
build-schedule [options]

-V --version            Give version information for this program
-h --help --usage       Describe usage of this program.
   --help-opt=OPTION    Give help information for a given option
-v --verbose[=VERBOSITY] Give information about what the program is
                        doing.  Set value to control what information
                        is given.
--quite -q --silent     Program should generate no output except in
                        case of error.
   --no-warn            Avoid issuing warnings about non-fatal
                        problems.

-l --url-list=FILENAME  File with complete list of URLs to schedule
-s --schedule=FILENAME  Override location of the schedule
-t --spread-time=SECONDS Time over which to spread checking; default 10
                       days
-S --start-offset=SECONDS Time offset from now for starting work (can
                        be negative)
-d --ignore-db          Set the time with no regard to curent setting
-i --ignore-link        Set the time with no regard to link status
   --no-warn            Avoid issuing warnings about non-fatal
                        problems.
   --config-file=FILENAME Load in an additional configuration file
```

Appendix G Packages Which Work With LinkController

LinkController uses several programs and can work with several others. This section covers the most important ones.

G.1 The CDB utilities

In order to have LinkController working you must have installed these. It is worth looking at the utilities that are provided, especially `cdbdump` which will let you look at the contents of the file. You should be aware that `cdbget` program which is provided *won't* be able to get at the full contents of the index files since they contain repeated keys.

More information on cdb and new releases can be got from the www page.

`http://cr.yp.to/cdb.html`

When using link controller you are advised to use FreeCDB which, because of its better license terms, has the extra guarantee that it will be possible for anybody to distribute fixed versions and provide support for them.

`http://packages.debian.org/freecdb`

G.2 The Tie-Transact-Hash Perl Module and Programmes

This is a Perl module written by myself which includes a program which allows direct examination and editing of Berkeley databases. It can be useful for debugging and correcting problems in the LinkController Link database or schedule file.

Tie::TransactHash can be downloaded from CPAN, the Comprehensive Perl Archive Network get there via:

`http://www.perl.com/language/info/software.html`

Appendix H Terms

H.1 Infostructure

An infostructure is a concept which was introduced in Link Checking in the MOMspider package. It is a collection of related resources. For us it's mostly just a way of saying 'web pages' but includes things like databases which may not have any real identifiable 'pages' that we can read through directly.

H.2 Link

The term link in LinkController is used for a connection between two resources. It's existence really comes from the 'class' or piece of type of computer data which is used to store information about 'links'. Properties of a link include:

H.3 Resource

A resource is almost anything. 'It' can range from a person to an HTML file to a computer to a database or presumably eventually to phone numbers, possibly physical hardware. This generality is a very important concept for the World Wide Web. Really the key thing about a resource is that it can be 'identified'. See Section H.5 [URLs], page 45, for more details.

H.4 URIs

A URI or 'Uniform Resource Identifier' is a more generic form of the URL Section H.5 [URLs], page 45 which also includes URNs Section H.6 [URNs], page 46. It also allows links to abstract objects which can't be reached through a network server. Since all URLs are URIs we mostly try to talk about URIs when we can since that includes both. Often people say URL when they mean URI. We try to use correct usage always so that in future we can support all forms of URI without confusing existing users. URIs and URLs are defined in RFC 2396.

H.5 URLs

A URL or 'Uniform Resource Locator' are the essence of the World Wide Web. Approximately, they are addresses through which 'resources' can be located. The idea is that almost anything can be given some kind of address in a form that a machine can work with. By defining a set of rules, this can then be converted into a URL. A URL has two parts.

The first tells us what rules to use and the second tells us what the address is. URLs and URIs are defined in RFC 2396. URLs are not the only kind of link, but they are the most common and currently the only ones LinkController really handles well.

H.6 URIs

A URN or 'Uniform Resource Name' is a URI Section H.4 [URIs], page 45 which is not a URL Section H.5 [URLs], page 45. This means a way of specifying a resource without saying how to get it. For example, a scheme which has been considered is for ISBN (International Standardised Book Numbers) numbers. This would allow us to specify a book as a resource but wouldn't tell us how to get it.

It's not totally clear where these will be useful in link checking (they are used internally in several computer systems), but LinkController intends to support them whenever needed, wherever possible.

- Knowing what the URL of the target resource of the connection is.
- Knowing whether the connection to the target resource has been working recently.
- Knowing when the connection to the target resource was last checked.

Within the programs, a link is different from a URL in that it is specifically aimed at checking connections, where a URL just specifies what the connection should be if it is working.

Program, Variable and File Name Index

This index includes all programs, variables and files.

$

~

B

C

D

E

F

I

L

M

P

R

S

T

Concept Index

R

S

U

V

W