Big Brother Manual
version 2.0.4

François Pottier

May 5, 2004

Contents

1 Introduction

1.1 What is Big Brother?

Big Brother allows you to keep an eye on your World Wide Web links. It is essential for webmasters and HTML authors who wish to make sure that the links contained in their pages are up-to-date; a chore which quickly gets out of hand without an automated tool! It is also invaluable to those who maintain a large list of bookmarks, since it can easily be used to check bookmark files.

Big Brother is distributed in source code, written in the Objective Caml language; it should compile "out of the box" on most Unix-based systems. Binaries for Windows 95/NT and RedHat Linux (i386) may also be available from the author's Web site. The whole program, including its source code, is placed in the public domain.

This version of Big Brother is a command-line tool. Putting a graphic interface—possibly written in a different programming language—on top of it should be reasonably easy. I no longer have time to do it, so if you're a programmer, think about it! Currently, the simplest way to use Big Brother is to sit down and write a one-line *script* (also known as *batch file*) to run Big Brother with appropriate options for your site. Once this is done, just run the script whenever you want to check your site.

In short, Big Brother...

- Verifies many links simultaneously, resulting in blazing speed.
- Supports remote (`http`) and local (`file`) links.
- Allows checking a site offline, through the use of *mappings*.
- Follows links recursively, allowing an entire site to be verified with a single command.
- Produces results either in a custom format (convenient for further processing by your own scripts), or directly as a human-readable HTML report.
- Supports the full HTML 4.0 language.
- Supports going through a proxy server.
- Allows controlling the frequency of network requests, to avoid overloading your server.
- Supports multiple password-protected *realms*.
- Has detailed documentation, available in HTML, PostScript and PDF formats.

Big Brother watches your documents and reports all mistakes—hence its name!

1.2 What Big Brother is *not*

When a webmaster removes a document from his site, he can either remove it for good, or replace it with a page containing a short explanatory message. Similarly, when he moves a document to a new location, he has a choice between configuring his Web server to provide clients with the new address (this is called automatic redirection), or replacing the document with a short page containing a link to the new location. In each of these two situations, if the first choice is made, then Big Brother can identify that the document is missing or has moved and alert you about it. If the second choice is made, there is no way for Big Brother to distinguish the explanatory message from the expected document, so the error will remain undetected.

Besides, one should point out that Big Brother is not an HTML syntax checker. Its job is to verify links, and most errors in your HTML documents will go unnoticed. This is a probably a good thing, since syntax checking is a difficult task, and an entirely different job. Big Brother is very lenient and accepts anything that looks remotely like HTML text.

Finally, Big Brother has a few limitations. The main one is that it does not support `ftp` links. This problem might be remedied in a future version.

1.3 Quick start

Big Brother might seem difficult to use at first glance, because it has so many command line options. However, I do not think it is a real problem. The reason is, you should typically use Big Brother to perform a few specific tasks (such as checking your Web site, if you're a webmaster) repeatedly. So, you can take a few minutes to look at Big Brother's options and write a script once and for all.

Several typical situations are described in section 3. Also, you should have at least a cursory glance at the Reference section (section 2), so you're aware of what Big Brother can do for you.

2 Reference

Big Brother is highly flexible. This section documents all of the options which can be specified on its command line. A short summary can be obtained by invoking `bigbro -help`.

2.1 General syntax

Big Brother is invoked as follows:

`bigbro` [*options*] url_1 url_2 ...

That is, following the program name, you have the ability to specify one or more options, and you must then specify a list of URLs. (This list can be empty; this makes sense if you use the `-stdin` option.) Options, as well as URLs, are separated by spaces.

URLs must be formed according to a certain syntax (see section 2.2). They can be absolute or relative. If they are relative, they shall be resolved with respect to the current directory. Under Unix, this means that you can specify file names instead of URLs and obtain the expected behavior. Under Windows, this does not work, because URLs use a slash as a separator, whereas file names use a backslash.

Big Brother checks each URL in the list. To specify how it should report the results, use the `-oraw` and `-ohtml` options.

Some options have no parameter. Others expect one parameter, which should come after the option's name, separated by a space. Parameters are usually integers, strings or regular expressions. The syntax of regular expressions follows that of Perl. (In fact, regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England.)

2.2 Syntax of URLs

This section gives some detail on the syntax of URLs, as used by Big Brother. These rules should be obeyed when supplying URLs to Big Brother.

There are two kinds of URLs: remote (`http`) and local (`file`) URLs. Remote URLs have the following syntax:

`http://` *server name* / *relative path*

Under Unix, local URLs are of the form

`file:///` *relative path*

Under Windows, they are of the form

`file://` *drive* / *relative path*

A *drive* is a letter followed by a colon. A *relative path* is a sequence of directory or file names, separated by slashes, and possibly ending with a slash.

These rules correspond to *absolute* URLs. A URL can also be *relative*, that is, depend on another URL. In that case, it only consists of a relative path, such as `../pics/bob.gif`. Note that some characters are illegal in URLs and must be encoded using a percent sign followed by their hexadecimal ASCII code. For instance, if a URL contains a space, it should be encoded as `%20`.

2.3 Setting up mappings: `-mapfrom` and `-mapto`

Mappings are a mechanism which allows URLs to be rewritten before being checked. The most typical use of mappings is to let Big Brother bypass a Web server and read documents directly from disk, as shown in the example below.

You can specify any number of mappings. All mappings are applied to the URL being checked, after it has been resolved (that is, turned into an absolute URL, if it was relative). A mapping is specified as follows:

`-mapfrom` regexp `-mapto` replacement

The mapping is applied by finding the first substring of the URL at hand which matches the specified regular expression, and replacing it with the replacement string. (If no substring matches the regular expression, the mapping has no effect.) The replacement string can contain `$1`, `$2`, etc; these sequences will be replaced by the text matched by the corresponding group in the regular expression. `$0` stands for the text matched by the whole regular expression.

Here is a simple and realistic example. Suppose I have a Web site available at `http://www.users.com/~tom/`, whose files are stored on my hard disk in the directory `/home/tom/web/`. When checking my site, I want Big Brother to read the documents directly off my disk, instead of requesting them from the Web server. So, I set up a mapping:

```
-mapfrom "^http://www\.users\.com/~tom/"
     -mapto "file:///home/tom/web/"
```

Now, suppose I ask Big Brother to check the URL `http://www.users.com/~tom/index.html`. The mapping applies, so the URL is rewritten and becomes `file:///home/tom/web/index.html`. Thus, Big Brother will read the file from disk, rather than request it from the server.

Let us explore this example a bit further. Assume the above index file contains a link to `../~amy/`. Amy's Web site is *not* stored on my hard disk. Will Big Brother be smart enough to request it from the server? Yes! Although Big Brother applies the mapping to a URL when trying to access it, it remembers the original URL and uses it as the base URL when resolving relative URLs. In slightly less technical terms, here is what this means: when it finds the relative link `../~amy/`, Big Brother resolves it. It is resolved with respect to the unmapped URL of the current document, which is `http://www.users.com/~tom/index.html`. So, the resolved URL is `http://www.users.com/~amy/`. At this point, the mapping is applied to this URL, but it does not match, so the URL remains unchanged. As a result, Big Brother properly sends a request to the Web server to retrieve this document.

So, to sum up, here's how to bypass a Web server using mappings. First, set up a mapping which maps `http` URLs to `file` URLs appropriately. (Have a look at the URL syntax rules in section 2.2.) Second, ask Big Brother to check the remote URL, as usual.

Now, here comes a more elaborate example, which shows how powerful mappings can be. I'm still Tom and I still have a Web site, but this time, only the HTML documents are stored on my hard disk – other files, such as images, are available only on the server. So, I want the mapping to apply only to HTML files. Here's the way to do it:

```
-mapfrom "^http://www\.users\.com/~tom/(.*\.html)$"
         -mapto "file:///home/tom/web/$1"
```

The regular expression matches only documents whose name ends with `.html`. Besides, the document name is enclosed by a *group* using `(` and `)`, which allows referring to it by `$1` in the replacement string. So, `http://www.users.com/~tom/index.html` is still turned into `file:///home/tom/web/index.html`, but URLs of image files, such as `http://www.users.com/~tom/tom.jpg`, are unaffected.

2.4 Controlling recursion: `-rec`

By default, recursion is off. This means that Big Brother reads the documents you explicitly specify (either on the command line or using the `-stdin` option), makes sure that the links they contain are valid, and stops.

If you want to check a whole site at once, you must turn recursion on. When recursion is on, Big Brother not only checks each link, but also determines whether it points to an HTML document. If it does, Big Brother fetches the document (which can be local or remote) and checks the links in it. If one of these links points to an HTML document, it is also fetched, and so on, recursively.

Of course, this process has to stop at some point, otherwise you are likely to check all of the World Wide Web! So, when recursion is on, you must provide a regular expression to define the boundary of your site. Every time Big Brother finds a link, it shall match it against the regular expression you have provided. If it does match, Big Brother fetches the document and checks it recursively; otherwise, it checks the link but doesn't follow it.

Here is the simplest, and most common, example. The address of my home page is `http://pauillac.inria.fr/~fpottier/`. If I want to check my whole site in a single run of Big Brother, I invoke it with the following option:

```
-rec "^http://pauillac\.inria\.fr/~fpottier/"
```

Since the expression doesn't end with a `$` sign, it matches all URLs which start with `http://pauillac.inria.fr/~fpottier/`. Notice that I inserted a backslash character in front of each dot, because otherwise a dot matches any character.

Here is another example: suppose I maintain several Web servers, but all of them are within the domain `orange.com`. Then, I can use the following regexp:

```
-rec "^http://.*\.orange\.com/"
```

This will match the address of any document located on any server within the domain `orange.com`.

Note that the address is checked against the recursion regexp *after* the mappings have been applied to the address. So, if you are using mappings, say, to map a remote site to a certain folder of your hard disk, the recursion regexp should describe the local files' addresses, not the remote ones'. For instance, using `^file:` as recursion regexp simply allows recursion on all local files.

2.5 Checking `-local` or `-remote` links

Links are divided into two categories: local (`file`) and remote (`http`) links. The former represent files or folders on your hard disk, while the latter can only be accessed by establishing a connection to a remote machine. Checking local links is usually much faster and does not require your machine to have networking capabilities.

You might want to ignore local links (say, because remote links are more likely to become invalid as time passes) or remote ones (say, because you do not want to use your modem).

Note that by default, neither local nor remote links shall be checked. You *must* specify `-local`, `-remote` or both if you actually want to get anything done.

2.6 Configuring local checks: `-index` and `-localhtml`

There are a few things Big Brother needs to know when checking local (`file`) links.

The first one is the default index file name. Whenever a local link points to a directory, Big Brother looks for an index file inside it. (The index file is the file that should be displayed instead of the directory listing. As its name implies, it is supposed to describe what's in the directory, but it does not have to. It is also all right if there is no index file at all.) If recursion is on, and if this file's URL matches the recursion regexp, then it shall be opened and checked. If you don't use the `-index` option, Big Brother behaves as if you had said

```
-index "index.html"
```

This setting affects local links only, because in the case of remote links, replacing the directory with its index file is done transparently by the remote server.

Additionally, Big Brother needs to be able to tell which files are HTML files. (This is necessary when recursion is on, because it would make no sense to download, say, an image file and to try analyzing it.) For remote files, there is no problem, because the server shall supply the necessary information. There is one, however, for local files, where no information is available. So, Big Brother needs you to supply a regular expression using the `-localhtml` option. When Big Brother finds a link to a local file, it determines whether it is an HTML file by matching its name against this regular expression. If it does match, the file is assumed to be an HTML document. If you don't use the `-localhtml` option, Big Brother behaves as if you had typed

```
-localhtml "\.s?htm"
```

which means that a file is considered as an HTML file if its name contains `.htm` or `.shtm`. Here is another example. My site contains files in English and in French, whose names end in `.html.en` and `.html.fr`, respectively. I can let Big Brother know about it by typing

```
-localhtml "\.html\.(en|fr)$"
```

2.7 Using a proxy: `-proxy` and `-noproxy`

If your machine is located behind a firewall which isolates it from the outside world, you might need to use a proxy. A proxy is a gateway machine which sits between you and the rest of the network and relays your machine's HTTP requests.

If you don't know whether you need a proxy, ask your nearby guru. Having a look at your browser's settings might also help. If you find that you need to use a proxy, give its name to Big Brother using the `-proxy` option. If the proxy uses a port number other than 80, you must include a colon followed by the port number at the end of the name. For instance, I use

```
-proxy www-rocq.inria.fr:8080
```

You usually don't need to use the proxy when connecting to a machine within the same organization as yours, because it is also behind the firewall. In that case, bypassing the proxy should be more efficient. You can tell Big Brother to do so by using the `-noproxy` option. This option is followed by a regular expression. The expression describes a set of addresses which can be accessed without using the proxy. For instance, I can access all machines within the domain `inria.fr` without using the proxy, so I specify

```
-noproxy "^http://.*\.inria\.fr/"
```

2.8 Controlling concurrency: `-maxthreads`

Big Brother is a *multi-threaded* program. This means that it can conduct several tasks in parallel. This allows it to gain a lot of time, because while it is waiting for a remote server to answer a request, it can still work on other tasks.

There is a limit to the number of tasks (called *threads*) that can be run concurrently. (If there wasn't, Big Brother would eagerly try to do hundreds of things at once and might use up all of your machine's resources.) By default, this limit is 16, as if you had typed

```
-maxthreads 16
```

which means that Big Brother will check up to 16 links simultaneously. You can experiment. Big Brother should use less processing power and memory if the limit is decreased, and it should be faster if the limit is increased.

2.9 Reading from a pipe: `-stdin`

By default, URLs to be checked are initially specified on the command line. However, if you add the option `-stdin` to the command line, then Big Brother will also read URLs from standard input. It expects to find one URL per line, without any other information.

This could be useful, for instance, if you have a file containing a raw list of URLs. You could have it checked by issuing the command

```
bigbro -stdin [other options] < myfile.raw
```

Big Brother will then read the file `myfile.raw`, line per line, and check each of the URLs it contains.

2.10 Measuring one's patience: `-timeout`

This option lets you specify how long, in seconds, Big Brother should wait before giving up on a network request. By default, there is no timeout, so Big Brother is infinitely patient. If you want Big Brother to give up after, say, 5 minutes, you should type

```
-timeout 300
```

This time limit concerns a whole network request: that is, it should allow enough time to locate the server, send the request and receive a full answer.

For technical reasons currently beyond my control, this feature is only partially supported in the Windows version. In some cases, Big Brother might still wait longer than allowed by this option.

2.11 Being `-gentle` with servers

This option lets you specify how long Big Brother should wait between two requests to the same server. By default, Big Brother is not gentle at all, so it will issue HTTP requests as quickly as it can. This could make the target Web server(s) noticeably less responsive. So, to avoid overload, you should specify how many seconds must elapse between consecutive requests to a single server, by using the `-gentle` option. A reasonable setting is

```
-gentle 5
```

Of course, this will cause link checking to take more time.

2.12 Using passwords: `-realm`, `-user` and `-password`

Some servers have private areas, where a *user name* and a *password* have to be supplied in order to download documents. Such areas are called *realms*. Big Brother is able to deal with them, provided that you tell it which documents require which user name & password combination. This is done as follows:

```
-realm regexp -user string -password string
```

The realm is described by a regular expression. As usual, a regular expression describes a set of addresses. That is, a document is considered as part of the realm if its address matches the regular expression. The user name and password are just strings. Here is an example:

```
-realm "^http://localhost/~francois/private/"
       -user francois -password bikini
```

You can use these three options more than once, to define several password-protected realms with different user names or passwords.

2.13 Reporting results: `-oraw` and `-ohtml`

Big Brother can display the results of its work in two different ways: either by printing "raw" information, or by generating a report in HTML format. The former is rather difficult to read for humans, but easy to parse for a machine. I shall document its format when I have some time. It is requested using `-oraw`. The latter is a simple HTML description of which links were checked and what the outcome was. It is requested using `-ohtml`.

Both `-oraw` and `-ohtml` expect an argument. If it is the keyword `stdout`, Big Brother will send the data to the standard output channel, which means that it will appear in your terminal window. Otherwise, the argument is assumed to be a file name; Big Brother creates the file and writes the output to it.

For example, to store an HTML report in the file `results.html` and to see the raw information on screen, type

```
-oraw stdout -ohtml results.html
```

You must use at least one of these two options, since otherwise no output at all would be generated.

When it's done checking, Big Brother displays some statistics. They appear both in raw and HTML outputs. For each kind of URLs (local and remote), it displays the following information: number of documents whose existence was checked—number of documents actually read or downloaded—approximate volume of data transferred. This data can be interesting to assess your site's size!

2.14 Focusing on `-failures`

Usually, most of the links in your site should be correct, and you should only be interested in the wrong ones. But the error diagnostics can be hard to spot if they're scattered among hundreds of successful ones. To solve this problem, just add

```
-failures
```

to Big Brother's options. When you do this, only invalid links are reported. This option affects both raw and HTML output.

2.15 Checking `-fragments`

Some URLs contain a `#` sign followed by a pointer to a specific spot in the document, like this:

```
http://www.someone.com/help.html#me
```

The final part of this URL, `#me`, is called a *fragment*. (Many people call it an *anchor*.) It is a reference to a specific place within the document `http://www.someone.com/help.html`. This fragment should be *defined*, to indicate which place is meant by this name. Fragments are usually defined using the `<A>` tag, like this:

```
<A NAME="me">some text</A>
```

They can also be defined by adding an `ID` attribute to any HTML tag.

Naturally, part of Big Brother's job is to make sure that fragments are correctly defined. However, this can use a lot of network bandwidth. Imagine what happens when Big Brother checks the above URL. First, it asks the server `www.someone.com` whether the document `help.html` exists. This is usually very fast, because the server simply answers "yes, it does" without actually transmitting it. However, if Big Brother wishes to make sure that `#me` is defined, then it has to download the whole document and look for the definition in it.

One should point out that Big Brother handles the problem in the smartest possible way. That is, it will never download a document twice, even in the most complex situations. Nevertheless, checking fragments could mean downloading a lot more data. (Have a look at the statistics displayed at the end of each report to see exactly how much more.) Because of this, checking fragments is an option. It is off by default; in that case, Big Brother will simply ignore anything that comes after the `#` sign. To turn it on, add

```
-fragments
```

to the command line.

2.16 Ignoring certain links: `-ignore`

If you use Big Brother regularly, you might notice that some links are consistently reported to be incorrect, even though they work when you try them out with your browser.

There are several possible reasons for this problem. The most common one is that Big Brother does not behave exactly like a browser, and because of this, it sometimes runs into server bugs. Here's why. A browser connects to the remote server and says, "send me the document". The server says "all right" and complies. Big Brother works differently. It doesn't ask for the document; instead, it asks "does the document exist?". Some (buggy) servers are not used to this kind of request, even though they should be, since it is part of the HTTP standard. They answer something which could be translated as "uh?". This is why Big Brother reports the link as invalid, even though it looks valid when using a browser. (In technical terms, browsers do GET requests, while Big Brother does HEAD requests, and the latter are often badly supported by server software.)

So, if you run into this problem, there is no way to fix it, except have the webmaster switch to some better server software. If that is beyond your control, you can use the `-ignore` option to have Big Brother ignore the link. This option expects an argument, a regular expression which describes a set of addresses to be ignored. For instance, I use

```
-ignore "^http://www\.imdb\.com/M/"
```

because the Internet Movie DataBase's CGI scripts don't handle HEAD requests properly.

For convenience, you can use `-ignore` several times in a single command.

2.17 Getting version information: `-v`

If you want to determine which version of Big Brother you are using, just type

```
bigbro -v
```

It will then issue a short banner containing miscellaneous information, including its version number.

3 Examples

This section offers a few simple, but hopefully typical, examples of Big Brother's usage.

3.1 Checking your bookmarks

Most browsers store your bookmarks in a special HTML file. The name and location of that file depend on your particular brand of operating system and of browser. When using Netscape Navigator under Unix, it is called `bookmarks.html` and located in a directory called `.netscape` inside your home directory. Under Windows NT, it might be `C:\Program Files\Netscape\Navigator\bookmark.htm`. If you are using another setup, you'll need to experiment to discover this information. (Let me know what you find out, and I'll add it to the manual.)

Which options to use? Since bookmarks usually point to external sites, you won't need any mappings. Recursion should be off, because you're only interested in knowing whether these documents exist, not whether the links they contain are valid. As always, we must avoid overloading servers, so we use `-gentle` to slow down Big Brother. To obtain human-readable results, we request Big Brother to store a report in HTML format using `-ohtml`. So, the command will look like this, under Unix:

```
bigbro              \
  -local            \
  -remote           \
  -gentle 5         \
  -ohtml report.html \
  -failures         \
  -fragments        \
  $HOME/.netscape/bookmarks.html
```

(The \ characters at the end of each line are used to indicate that this is a single command, even though it is written on several lines for clarity.) You can store this command in a script to avoid typing it again and again. Also, it is easy to have the report mailed to you and the script run every week. Under Windows, your batch file might look like this:

```
bigbro              \
  -local            \
  -remote           \
  -gentle 5         \
  -ohtml report.htm  \
  -failures         \
  -fragments        \
  file://C:/Program%20Files/Netscape/Navigator/bookmark.htm
```

Note that I had to replace the space in "`Program Files`" with an escape sequence "`%20`", because spaces are illegal in URLs.

Of course, your situation might be slightly different. For instance, remember to specify a proxy if you need one. Also, if your bookmarks contain pointers to password-protected sites, you will have to give the password to Big Brother by using `-realm`, `-user` and `-password`. (Under Windows NT or Unix, you might then want to make the script unreadable to other users, so as to protect your privacy.)

3.2 Checking your site

This section shows which command I use to check my own site. You should be able to adapt it easily to your case. First, here is the command:

```
bigbro                                                  \
  -mapfrom "^http://pauillac\.inria\.fr/~fpottier/" \
  -mapto file://$HOME/public_html/                  \
  -rec "^file:"                                     \
  -local -remote                                    \
  -proxy www-rocq.inria.fr:8080                     \
  -noproxy "^http://.*\.inria\.fr/"                 \
  -timeout 600                                      \
  -gentle 5                                         \
  -oraw stdout                                      \
  -ohtml report.html                                \
  -failures                                         \
  -fragments                                        \
  -ignore "^http://www\.imdb\.com/M/"               \
  http://pauillac.inria.fr/~fpottier/
```

(The \ characters at the end of each line are used to indicate that this is a single command, even though it is written on several lines for clarity.)

Here is an explanation of the options used above. First, I define a mapping (see section 2.3), which tells Big Brother that any document belonging to my site can be read directly from the `public_html` subdirectory of my home directory. (Using `$HOME` in this way is Unix-specific, but you can specify a full path explicitly under Windows.) Then, I enable recursion (see section 2.4) within my site. Determining whether a file belongs to my site is easy, since if it does, then it resides on disk, so it has a `file:` URL. This explains why I used `-rec "^file:"`. Next, I enable checking both remote and local links. I then let Big Brother know about my proxy (note that the proxy's name ends with a custom port number which comes after a colon). The proxy is unnecessary when accessing machines within the domain `inria.fr`, hence the use of the `-noproxy` option. Next, I set the timeout value to 10 minutes and, to avoid consuming too much server time, I specify that at least 5 seconds should elapse between two requests to the same server. (This is especially important when using a proxy, since nearly *all* requests are sent to the proxy.) Next, I request "raw" output onscreen and human-readable output to a file called `report.html`. Displaying failures only saves time and makes the report more readable. I want fragments (see section 2.15) to be checked. I use `-ignore` (see section 2.16) to avoid checking some URLs which I know will cause failures. Finally, I tell Big Brother where to start by specifying the main URL for my site.

That's it! It might seem overwhelming at first sight, but remember that once the command has been stored in a script, all that's needed is to run the script whenever you want your site verified.

4 Miscellaneous

4.1 Disclaimer

This software was written between 1996 and 2001 by François Pottier. The author has made a substantial effort to ensure that it is bug free, but there is no such thing as perfect software. No warranty whatsoever, either implied or express, is made concerning its correct functioning. You, the user, assume all responsibility for any damage caused, directly or indirectly, by its use.

4.2 Licence

This software, including its source code, is placed in the public domain. You may use, modify, or distribute it without any restrictions. You may use it as part of your own software projects, be they for profit or not. You may include it in software collections, including Linux distributions. Of course, if you do anything interesting with it, I would like to know about it and possibly receive some credit, but that is not a requirement.

The reason I place Big Brother in the public domain is mainly lack of time. Still, I do consider it a high-quality product. Version 2.0.4 hasn't been tested much, so it probably contains a few bugs, but I am sure it has great potential. It is written in Objective Caml—an uncommon language, but definitely worth learning. It alone is the main reason why this program is so robust. Trust me!

4.3 Compiling Big Brother

To compile Big Brother from its source code, you need

- some version of GNU `make`;
- the Objective Caml compiler, version 3.07 or later (`http://caml.inria.fr/`).
- Markus Mottl's O'Caml interface to Philip Hazel's PCRE library (`http://pcre-ocaml.sourceforge.net/`).

To build Big Brother, issue the following commands:

```
touch .generated-dependencies
make depend
make
```

If you should wish to also compile Big Brother's documentation, you need

- a TeX compiler, with a LaTeX2e distribution;
- a copy of the `hyperlatex` package, version 2.5 or later, by Otfried Cheong;
- a copy of `ps2pdf`, part of Aladdin Ghostscript.

4.4 Contacting me

If you want to talk to me, suggest enhancements or complain about bugs, feel free to contact me. My email address is `Francois.Pottier@inria.fr`.