

## Web Spider

### Documentation

The working procedure of the spider is simple. Following are the steps the spider follows:

1. Get the starting URL. Append it to queue.
2. Extract a link from queue. If no more link, go to step 9.
3. Visit the link. Increment the count of number of links visited.
4. For all internal link found in the page visited, do  
    append link to queue  
    add link to currently visiting link's adjacency list
5. Assign a unique incremental ID to the visited link.
6. Insert the link to a 'visited' search tree so that we can query later and that we don't visit a link twice.
7. Append <ID> <URL> to queue file.
8. Go to Step 2
9. Allocate a matrix with number of rows and columns equal to the number of visited links.
10. Build the matrix in memory.
11. Dump matrix to file.
12. Exit.

#### **How the matrix is built:**

1. When a link is visited, each time an internal link is found, it is inserted to the adjacency list of the visited link.
2. When all the internal links are extracted from page, a unique ID is computed for this link and assigned to this link. The whole information is then inserted into a binary search tree. Each node of the binary search tree contains following fields:  
    int id; /\* unique ID for the URL \*/  
    char \*url; /\* the link \*/  
    list\_t adj; /\* adjacency list of the visited link \*/
3. The <ID> <URL> entry is then appended to the queue file.
4. The matrix is built recursively by traversing nodes of the tree. Row and Column are calculated from the ID of parent URL and each internal link in the adjacency list.
5. Row and Column in the matrix are according to the queue file.

### **How URL checking is done:**

1. When a link is found, it is checked whether the link is from the originating web-site.
2. If the link is found internal to the web-site, it is matched against a regular expression pattern that we allow to be a valid link. In our program I used:

`(/|\.htm|\.php(\?.*)?)$`

'\$' matches to the end of link. So here we're allowing only links ending with '/', '.htm' and '.php' with optional '?name=value' query string after '.php'. We consider it to be allowed only if matches the regular expression.

### **Explanation of the Code**

#### **File Listing:**

##### **bst.h, bst.c:**

bst.c contains implementation of binary search tree. The function names are self-explanatory.

##### **list.h, list.c:**

list.c contains implementation of linked-list data structure and their operations

##### **queue.h, queue.c:**

queue.c contains implementation of Queue data structure and their operations

##### **mygetlinks.h, mygetlinks.c:**

mygetlinks.c contains functions for fetching links from a given URL. It uses 'libwww' library for robust visiting of URLs and very much fault tolerant.

##### **myregex.h, myregex.c:**

myregex.c contains functions for regular expression matching. Right now we need only one function and it is defined as macro in myregex.h.

##### **mypider.h, myspider.c:**

mypider.c contains the main program. It uses other above source files for doing its job.

## **Flow of the program:**

mypider.h includes necessary header files and data structures for the program. It defines following structure:

```
struct nodeinfo_struct {
    int id;
    char *url;
    list_t adj;
};
typedef struct nodeinfo_struct nodeinfo_t;
```

Each visited link contains above information.

**id:** a unique id assigned to the URL. We will refer to the URL with this id in the matrix.

The queue file will contains lines of <id> <URL>

**url:** the visited link

**adj:** the adjacency list of the visited link. The list contains all the internal links of the visited link.

All the necessary global variables are declared at the top of the 'mypider.c' file.

Execution starts from main() function.

I'm referring to line numbers of 'mypider.c' so that you can match the code with my talk and can understand flow of control of the program.

**Line# 57:** `baseurl = get_hostname(argv[1]);`

This statement finds out the URL of the originating host. For example, if spider is told to crawl <http://www.website.com/internal/>, get\_hostname() will return '<http://www.website.com/>'. Each time a link is found, the link is checked whether it contains baseurl at the starting. If it doesn't contain baseurl at the starting, then it's external link.

**Line# 63:**

`qinsert(&global_queue, strdup(argv[1])); /* link to be visited */`

This statement appends the link to crawl to the queue.

**Line# 69:**

`init_spider();`

init\_spier() initializes the program by doing various initializing tasks. See the function's body for more information.

**Line# 73:**

```
if (nlinks != INFINITE)
    fprintf(stderr, "\t*** Maximum %d links will be visited\n", nlinks);
```

If we pass an extra argument to the program saying how many links to visit, 'nlinks' contains it. INFINITE is defined as -1 in the file.

Now the program is going to enter a loop. The loop does the following:

**Line# 76:**

```
while ((url = qextract(&global_queue)) != NULL) {
```

While there's link to visit in the queue, it is extracted from queue.

**Line# 77:**

```
if (link_visited(url) == false) {
```

If the link is not yet visited. We're not going to visit a page twice. Whenever a link is visited, it is inserted into a binary search tree. link\_visited() searches the link in the tree and returns 'true' if found.

Inside the above condition the whole link crawling happens.

We increment 'count' and 'global\_count'. 'count' is just for checking how many links we already visited. 'global\_count' is equal but used for building matrix. Matrix will contain 'global\_count' number of rows and columns.

**Line# 84:**

```
_getlinks(url);
```

It is defined in 'mygetlinks.c' file. It crawls the URL and finds out all the internal links from it. After it returns we're going to build a node for inserting to the binary search tree. We allocate with 'malloc()'. After allocating, we set the node fields. get\_new\_id() returns an incremental unique id [integer value]. So, each visited link gets a unique ID.

**Line# 93:**

```
global_tree = tree_insert(global_tree, ninfo, nodeinfo_urlcmp);
```

The node is inserted into binary search tree. After this line, the <id> <URL> entry is written to the queue file.

The above things happen till there are links in the queue, or our limit of links to visit [if we specify as 2<sup>nd</sup> argument to program] exceeds.

Line# 113:

```
global_matrix = malloc(global_count * sizeof(int *));  
for (i = 0; i < global_count; i++) {  
    global_matrix[i] = calloc(1, global_count * sizeof(int));  
}
```

The matrix is allocated here. NxN matrix. N = global\_count = # of links visited by 'getlinks(url)'. A two dimensional array initialized by above statements.

**Line# 119:**

```
build_matrix(global_tree);
```

The matrix is built from the information found in the binary search tree.

**Line# 123:**

```
dump_matrix(MFILE);
```

Dumps the matrix to 'matrix.txt' [#define MFILE "matrix.txt"]

**Line# 126:**

```
destroy_spider();
```

Destroys the spider before exiting.

**Note:** During call to 'getlinks(url);' whenever a new internal link is found, do\_something\_with\_link() is called for each. In this function the link is checked whether it contains the 'baseurl' [the main website]. In this function we also check for existence of '#' in the link. Because links with it specify some paragraph of the page. For example,

<http://www.web.com/test/index.html#zzz>

<http://www.web.com/test/index.html#kkk>

Both links specify same file. So we check that for no confusion and so that we don't get fooled by it and visit the same link twice.

Finally in this function we call 'match\_regex()'. It checks whether the link contains the pattern we expect the link to be like. If matched [returning 0], we insert it to the queue for visiting. Also it is inserted to the adjacency list of the currently visiting link, since it's internal link of the link being visited right now.

'init\_libwww()' function is defined in 'mygetlinks.c'. It used libwww specific things to initialize the 'libwww' things.

## **Compiling and Using the program:**

1. Detailed way of compiling and using the program is described in the INSTALL file. You will get it in the archive of the source code. I'm pasting it here too.

==INSTALL==

Requirements:

=====

1. 'libwww' [<http://www.w3.org/Library/>] library.  
You can get it from:  
<http://www.w3.org/Library/Distribution.html>  
<http://www.idm.ru/content/view/9/8/lang,en/>

Installing w3c-libwww [if not already installed]:

=====

Get the 'w3c-libwww' library source code from following:  
<http://www.w3.org/Library/Distribution.html#Tar>  
<http://www.w3.org/Library/Distribution/w3c-libwww-5.4.0.tgz>  
Build the library. You will get how to build it from:  
<http://www.w3.org/INSTALL.html#Unix>

In short you need to follow the steps:

- i. Copy the w3c-libwww-XXX.tgz to some folder
- ii. In your terminal, 'cd /path/to/where/you/saved'
- iii. tar -xzf w3c-libwww-XXX.tgz
- iv. cd w3c-libwww
- v. ./configure
- vi. make
- vii. make install

Now you have 'libwww' library installed in system.

NOTE: To access Windows Drives , you need to follow directory names like following in cygwin terminal [shell]:

D:     => /cygdrive/d  
C:     => /cygdrive/c  
....

For example, to go to D:\test folder  
cd /cygdrive/d/test

Everything else is like Unix environment.

Building Spider:

=====

In UNIX:

=====

1. Open a terminal. Go to the directory where the archive is
2. Extract the archive  
tar -xzf myspider-1.0.tar.gz  
Or you can extract using other tools you have.  
For myspider-1.0.zip file,  
unzip myspider-1.0.zip
3. Change to the extracted directory  
cd myspider-1.0
4. Build the program  
make
5. Run the program  
./myspider http://www.website.com/  
To Visit only 20 links, do  
./myspider http://www.website.com/ 20

Two files will be produced.

1. 'queue.txt' contains the queue of links
2. 'matrix.txt' contains the connectivity matrix

Note that 'matrix.txt' uses ID for each Link. The ID->URL can be found in 'queue.txt'.

In Windows [Using Cygwin Environment]:

=====

1. You will have to get 'cygwin' and install it in your system. You will get it from following:  
http://www.cygwin.com/  
http://www.cygwin.com/setup.exe

During software selection please install software development

tools as much as possible when selection window arrives. 'Autoconf', 'Automake', 'Libtool', 'gcc' [compiler is must].

After installing you'll see a shortcut icon of cygwin in your desktop. It is available from Start menu also. Also where you installed it [default is c:\cygwin].

2. Extract files from the spider archive. You may use WinZip to extract.

3. Now build the spider for Windows. First three steps are same as UNIX.

4. make -f Makefile.cygwin

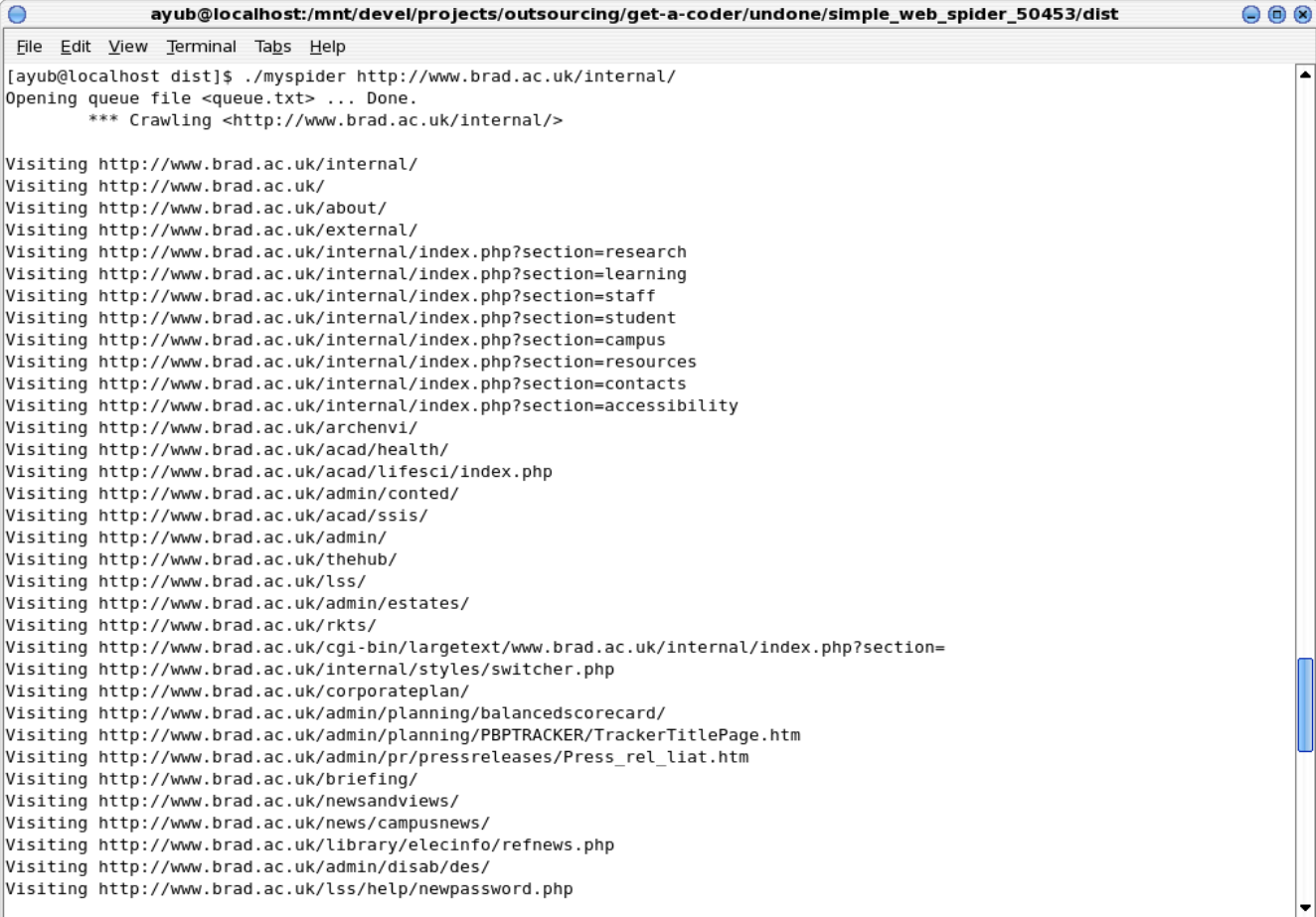
5. ./myspider <http://www.website.com/somedir/>

Thanks  
Ayub  
TechnoVilla

**Spider in action:**




## Screenshot of spider crawling...



```
ayub@localhost:/mnt/devel/projects/outsourcing/get-a-coder/undone/simple_web_spider_50453/dist
File Edit View Terminal Tabs Help
[ayub@localhost dist]$ ./myspider http://www.brad.ac.uk/internal/
Opening queue file <queue.txt> ... Done.
*** Crawling <http://www.brad.ac.uk/internal/>

Visiting http://www.brad.ac.uk/internal/
Visiting http://www.brad.ac.uk/
Visiting http://www.brad.ac.uk/about/
Visiting http://www.brad.ac.uk/external/
Visiting http://www.brad.ac.uk/internal/index.php?section=research
Visiting http://www.brad.ac.uk/internal/index.php?section=learning
Visiting http://www.brad.ac.uk/internal/index.php?section=staff
Visiting http://www.brad.ac.uk/internal/index.php?section=student
Visiting http://www.brad.ac.uk/internal/index.php?section=campus
Visiting http://www.brad.ac.uk/internal/index.php?section=resources
Visiting http://www.brad.ac.uk/internal/index.php?section=contacts
Visiting http://www.brad.ac.uk/internal/index.php?section=accessibility
Visiting http://www.brad.ac.uk/archenvi/
Visiting http://www.brad.ac.uk/acad/health/
Visiting http://www.brad.ac.uk/acad/lifesci/index.php
Visiting http://www.brad.ac.uk/admin/conted/
Visiting http://www.brad.ac.uk/acad/ssis/
Visiting http://www.brad.ac.uk/admin/
Visiting http://www.brad.ac.uk/thehub/
Visiting http://www.brad.ac.uk/lss/
Visiting http://www.brad.ac.uk/admin/estates/
Visiting http://www.brad.ac.uk/rkts/
Visiting http://www.brad.ac.uk/cgi-bin/largetext/www.brad.ac.uk/internal/index.php?section=
Visiting http://www.brad.ac.uk/internal/styles/switcher.php
Visiting http://www.brad.ac.uk/corporateplan/
Visiting http://www.brad.ac.uk/admin/planning/balancedscorecard/
Visiting http://www.brad.ac.uk/admin/planning/PBPTRACKER/TrackerTitlePage.htm
Visiting http://www.brad.ac.uk/admin/pr/pressreleases/Press_rel_liat.htm
Visiting http://www.brad.ac.uk/briefing/
Visiting http://www.brad.ac.uk/newsandviews/
Visiting http://www.brad.ac.uk/news/campusnews/
Visiting http://www.brad.ac.uk/library/elecinfo/refnews.php
Visiting http://www.brad.ac.uk/admin/disab/des/
Visiting http://www.brad.ac.uk/lss/help/newpassword.php
```

Screenshot of spider crawling 20 links [I used 20 to show the output in a screen]. A matrix of 20x20 will be generated from the crawling. A queue file of 20 lines will be generated from the command below.

A screenshot of a terminal window titled 'ayub@localhost:/mnt/devel/projects/outourcing/get-a-coder/undone/simple\_web\_spider\_50453/dist'. The terminal shows the execution of a script named 'myspider'. The user enters the command './myspider http://www.brad.ac.uk/internal/ 20'. The script outputs 'Opening queue file <queue.txt> ... Done.', followed by status messages: '\*\*\* Maximum 20 links will be visited' and '\*\*\* Crawling <http://www.brad.ac.uk/internal/>'. It then lists 20 URLs being visited, including various internal and external links of brad.ac.uk. After the list, it says 'Done.', 'Building matrix ... Done.', and 'Writing matrix to <matrix.txt> ... Done.'. The prompt returns to the user: 'ayub@localhost dist]\$'.

```
ayub@localhost:/mnt/devel/projects/outourcing/get-a-coder/undone/simple_web_spider_50453/dist
File Edit View Terminal Tabs Help
[ayub@localhost dist]$ ./myspider http://www.brad.ac.uk/internal/ 20
Opening queue file <queue.txt> ... Done.
*** Maximum 20 links will be visited
*** Crawling <http://www.brad.ac.uk/internal/>

Visiting http://www.brad.ac.uk/internal/
Visiting http://www.brad.ac.uk/
Visiting http://www.brad.ac.uk/about/
Visiting http://www.brad.ac.uk/external/
Visiting http://www.brad.ac.uk/internal/index.php?section=research
Visiting http://www.brad.ac.uk/internal/index.php?section=learning
Visiting http://www.brad.ac.uk/internal/index.php?section=staff
Visiting http://www.brad.ac.uk/internal/index.php?section=student
Visiting http://www.brad.ac.uk/internal/index.php?section=campus
Visiting http://www.brad.ac.uk/internal/index.php?section=resources
Visiting http://www.brad.ac.uk/internal/index.php?section=contacts
Visiting http://www.brad.ac.uk/internal/index.php?section=accessibility
Visiting http://www.brad.ac.uk/archenvi/
Visiting http://www.brad.ac.uk/acad/health/
Visiting http://www.brad.ac.uk/acad/lifesci/index.php
Visiting http://www.brad.ac.uk/admin/conted/
Visiting http://www.brad.ac.uk/acad/ssis/
Visiting http://www.brad.ac.uk/admin/
Visiting http://www.brad.ac.uk/thehub/
Visiting http://www.brad.ac.uk/lss/
Visiting http://www.brad.ac.uk/admin/estates/
Done.
Building matrix ... Done.
Writing matrix to <matrix.txt> ... Done.
ayub@localhost dist]$
```

## **QUEUE FILE**

```
===== queue.txt =====  
0 http://www.brad.ac.uk/internal/  
1 http://www.brad.ac.uk/  
2 http://www.brad.ac.uk/about/  
3 http://www.brad.ac.uk/external/  
4 http://www.brad.ac.uk/internal/index.php?section=research  
5 http://www.brad.ac.uk/internal/index.php?section=learning  
6 http://www.brad.ac.uk/internal/index.php?section=staff  
7 http://www.brad.ac.uk/internal/index.php?section=student  
8 http://www.brad.ac.uk/internal/index.php?section=campus  
9 http://www.brad.ac.uk/internal/index.php?section=resources  
10 http://www.brad.ac.uk/internal/index.php?section=contacts  
11 http://www.brad.ac.uk/internal/index.php?section=accessibility  
12 http://www.brad.ac.uk/archenvi/  
13 http://www.brad.ac.uk/acad/health/  
14 http://www.brad.ac.uk/acad/lifesci/index.php  
15 http://www.brad.ac.uk/admin/conted/  
16 http://www.brad.ac.uk/acad/ssis/  
17 http://www.brad.ac.uk/admin/  
18 http://www.brad.ac.uk/thehub/  
19 http://www.brad.ac.uk/lss/  
20 http://www.brad.ac.uk/admin/estates/  
=====
```

## **CONNECTIVITY MATRIX FILE**

```
===== matrix.txt =====
```

\*\*\*\*\* CONNECTIVITY MATRIX \*\*\*\*\*

ID is according to the queue file

[illegible]