WebTree

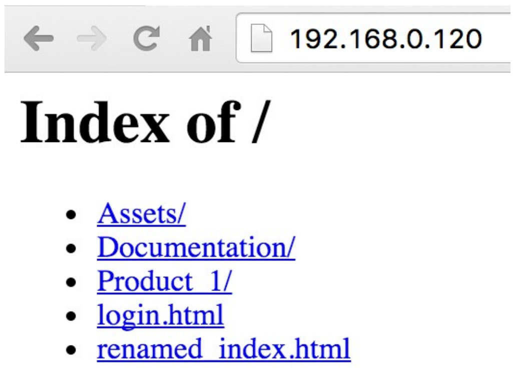
Stephan '5' Warren
github: codeaperature
https://www.linkedin.com/in/stephanwarren

October 25, 2015

Linux has the tree command that displays a files and directories in a organized hierarchical tree. Here is an example:

```
$ tree
.
├── Assets
│   ├── corwrprHm.gif
│   ├── Custom401.html
│   ├── Custom404.html
│   ├── EULA.css
│   ├── ftrwrprBg.gif
│   ├── hdrnavBg.gif
│   └── logo.png
├── Documentation
│   ├── Product_1.pdf
│   └── Product_2.pdf
├── login.html
├── Product_1
│   ├── Android
│   │   ├── 1.0.0
│   │   │   ├── 3rdpartytools
│   │   │   │   └── ndk_r10a.tar.gz
│   │   │   ├── chickenscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.2.1
│   │   │   ├── 3rdpartytools
│   │   │   │   └── ndk_r10d.tar.gz
│   │   │   ├── chickentratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.2
│   │       ├── 3rdpartytools
│   │       │   └── ndk_r10e.tar.gz
│   │       ├── chickenscratch.tar.gz
│   │       └── ReadMe.txt
│   ├── FireOS
│   │   ├── 1.0.1
│   │   │   ├── catscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.5.2
│   │   │   ├── catscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.0
│   │       ├── catscratch.tar.gz
│   │       └── ReadMe.txt
│   ├── iOS
│   │   ├── 1.1.1
│   │   │   ├── itchyscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.2.0
│   │   │   ├── itchyscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.3
│   │       ├── itchyscratch.tar.gz
│   │       └── ReadMe.txt
│   └── LinuxARM
│       ├── 2.0.2
│       │   ├── 3rdpartytools
│       │   │   └── cross_compiler_1.1.0.tar.gz
│       │   ├── bloodyscratch.tar.gz
│   │   └── ReadMe.txt
│   ├── 3.0.1
│   │   ├── 3rdpartytools
│   │   │   └── cross_compiler_2.0.0.tar.gz
│   │   ├── bloodyscratch.tar.gz
│   │   └── ReadMe.txt
│   └── 4.0.0
│   ├── 3rdpartytools
│   │   └── cross_compiler_5.0.0.tar.gz
│   ├── bloodyscratch.tar.gz
│   └── ReadMe.txt
├── Product_2
│   ├── Android
│   │   ├── 1.0.0
│   │   │   ├── 3rdpartytools
│   │   │   │   └── ndk_r10a.tar.gz
│   │   │   ├── chickenscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.2.1
│   │   │   ├── 3rdpartytools
│   │   │   │   └── ndk_r10d.tar.gz
│   │   │   ├── chickentratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.2
│   │   │   ├── 3rdpartytools
│   │   │   └── ndk_r10e.tar.gz
│   │   ├── chickenscratch.tar.gz
│   │   └── ReadMe.txt
│   ├── FireOS
│   │   ├── 1.0.1
│   │   │   ├── catscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.5.2
│   │   │   ├── catscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.0
│   │   ├── catscratch.tar.gz
│   │   └── ReadMe.txt
│   ├── iOS
│   │   ├── 1.1.1
│   │   │   ├── itchyscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   ├── 1.2.0
│   │   │   ├── itchyscratch.tar.gz
│   │   │   └── ReadMe.txt
│   │   └── 2.0.3
│   │   ├── itchyscratch.tar.gz
│   │   └── ReadMe.txt
│   └── LinuxARM
│   ├── 2.0.2
│   │   ├── 3rdpartytools
│   │   │   └── cross_compiler_1.1.0.tar.gz
│   │   ├── bloodyscratch.tar.gz
│   │   └── ReadMe.txt
│   ├── 3.0.1
│   │   ├── 3rdpartytools
│   │   │   └── cross_compiler_2.0.0.tar.gz
│   │   ├── bloodyscratch.tar.gz
│   │   └── ReadMe.txt
│   └── 4.0.0
│   ├── 3rdpartytools
│   │   └── cross_compiler_5.0.0.tar.gz
│   ├── bloodyscratch.tar.gz
│   └── ReadMe.txt
└── renamed_index.html
48 directories, 71 files
$
```

There may be some conditions where the tree's top level directory (TLD) is paired with the root directory of a web server. For example:


192.168.0.120

Index of /

- Assets/
- Documentation/
- Product_1/
- login.html
- renamed_index.html

In such a situation, the directories seen may differ just like the references above depending your htgroup, htpasswd and httpd.conf files. (These files can be modified to block or allow user's authorization to some directories & files such as Product_2 which is in the tree above, but not seen in the web page.) The exercise is now how can we determine which directories are allowed (foretelling which are blocked) in the web view served up with the Apache httpd service and the user's web browser.

This is where a client-side tool can be useful to help understand what each web user is authorized to access. To this effect, WebTree displays a similar-to-the-Linux tree that represents the user's access to directories and files.

Webtree:

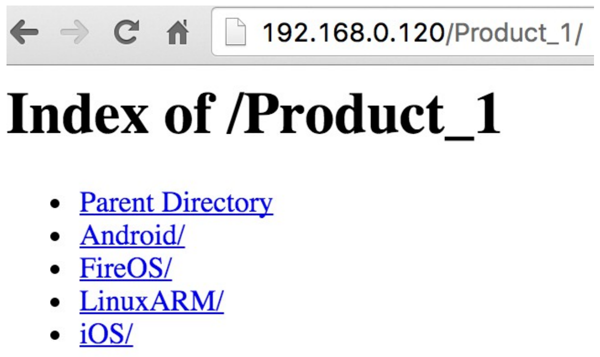
WebTree was designed to investigate URI passed as a tree's root directory allowed to accessible to a specific user. The URI is then recursed, “crawled” or “spidered” to find the directories (and files) under the URI. Cases where server-side read access is not allowed on files results in a 403 error on the client-side can be checked. Optionally, files can be saved to rebuild the tree or a local facsimile can be made.

Design:

If the program grabbed each page and file (head) in series, the tree build time would be the return time of each URL. Given latencies in http/https requests, Golang's goroutines are the only route I know to take. Going levels deep with recursive goroutines still can leave a user wondering whether the program is still running. WebTree handles the user interface in a “show the node as you have the node” manner. Assigning each node a wait task that blocks until the prerequisite node has completed before doing it's final dump of information. Other parts of WebTree accept channel messages in order to re-use http connections, save the tree and build a tree facsimile. Recursive death was adverted by mapping URIs already called (as to avoid returning back through the already mapped nodes).

Usage:

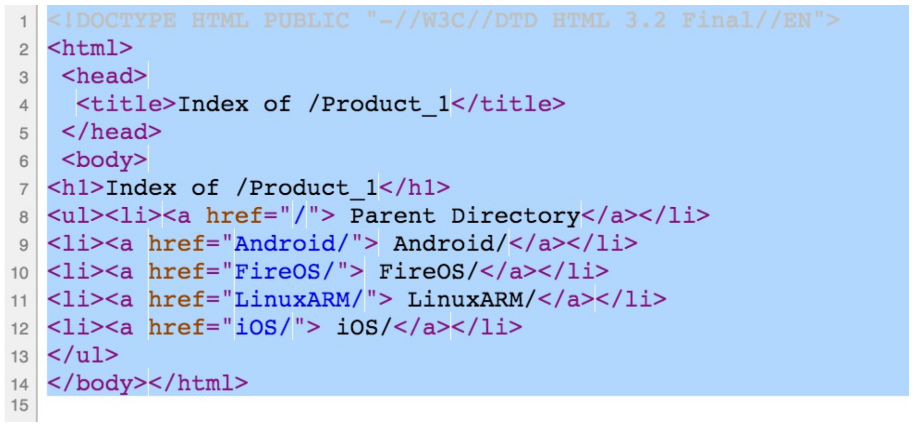
Most of the options in this example have been documented & tested. One of the sub-directories under the the continuing example is:


192.168.0.120/Product_1/

Index of /Product_1

- Parent Directory
- Android/
- FireOS/
- LinuxARM/
- iOS/

Where the page source appears as:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /Product_1</title>
 </head>
 <body>
<h1>Index of /Product_1</h1>
<ul><li><a href="/"> Parent Directory</a></li>
<li><a href="Android/"> Android/</a></li>
<li><a href="FireOS/"> FireOS/</a></li>
<li><a href="LinuxARM/"> LinuxARM/</a></li>
<li><a href="iOS/"> iOS/</a></li>
</ul>
</body></html>
```

Here the active URIs to recurse are found by a *simple* Xpath of :

/html/body/ul/li/a

Running the program without flags gives back the CLI instructions.

```
neon:WebTree stephan$ ./WebTree

Usage of WebTree:
  WebTree flags URI [user [password]]

Note:
  No user or password is required, but when user is set, a password can
  be provided as plain-text on the command line or via security prompt.

Flags:
  -clientpool int
        Maximum HTTP(S) Clients To Use (default 32)
  -pass string
        Password ... If HTTP(S) Authentication (default "(none)")
  -replicate
        Replicate Dir Struct With Touched Files
  -savetree string
        Save Tree in filename (default "(none)")
  -showdups
        Show Duplicate URIs
  -showfiles
        Show Files - Only Show Dir Tree
  -showstats
        Show Count of Files, Directories, Duplicates
  -testfiles
        Check That Files Are Able To Be Read
  -timeout-http int
        HTTP(S) Timeout For Clients To Use (secs) (default 30)
  -timeout-keepalive int
        KeepALive Duration For Clients To Use (secs) (default 10)
  -timeout-tls int
        TLS Timeout For Clients To Use (secs) (default 60)
  -timer
        Show Elapsed Time
  -user string
        User ID ... If HTTP(S) Authentication (default "(none)")
  -xpath string
        XPath to find hrefs (default "/html/body/ul/li/a")


neon:WebTree stephan$
```

Building and running with some of the CLI *parameters* yields:

```
/usr/local/go/bin/go build -i [/Users/stephan/golang/src/WebTree]
Success: process exited with code 0.
/Users/stephan/golang/src/WebTree/WebTree -showfiles -showstats -showdups -timer -testfiles
http://192.168.0.120 [/Users/stephan/golang/src/WebTree]
http://192.168.0.120/
ا ، صAssets/
، اص ، ص/ (VISITED)
، اص ، صCustom401.html
، اص ، صCustom404.html
، اص ، صEULA.css
، اص ، صcorwrprHm.gif
، اص ، صftrwrprBg.gif
، اص ، صhdrnavBg.gif
، صٌ ، صlogo.png
ا ، صDocumentation/
، اص ، ص/ (VISITED)
، اص ، صProduct_1.pdf
، صٌ ، صProduct_2.pdf
ا ، صProduct_1/
، اص ، ص/ (VISITED)
، اص ، صAndroid/
، صر اص ، ص/Product_1/ (VISITED)
، صر اص ، ص1.0.0/
، صر صر اص ، ص/Product_1/Android/ (VISITED)
، صر صر اص ، ص3rdpartytools/
، صر صر صر اص ، ص/Product_1/Android/1.0.0/ (VISITED)
، صر صر صر صٌ ، صndk_r10a.tar.gz
، صر صر اص ، صReadMe.txt
، صر صر صٌ ، صchickenscratch.tar.gz
، صر اص ، ص1.2.1/
، صر صر اص ، ص/Product_1/Android/ (VISITED)
، صر صر اص ، ص3rdpartytools/
، صر صر صر اص ، ص/Product_1/Android/1.2.1/ (VISITED)
، صر صر صر صٌ ، صndk_r10d.tar.gz
، صر صر اص ، صReadMe.txt
، صر صر صٌ ، صchickentratch.tar.gz
، صر صٌ ، ص2.0.2/
، صر اص ، ص/Product_1/Android/ (VISITED)
، صر اص ، ص3rdpartytools/
، صر صر اص ، ص/Product_1/Android/2.0.2/ (VISITED)
، صر صر صٌ ، صndk_r10e.tar.gz
، صر اص ، صReadMe.txt
، صر صٌ ، صchickenscratch.tar.gz
، اص ، صFireOS/
، صر اص ، ص/Product_1/ (VISITED)
، صر اص ، ص1.0.1/
، صر صر اص ، ص/Product_1/FireOS/ (VISITED)
، صر صر اص ، صReadMe.txt
، صر صر صٌ ، صcatscratch.tar.gz
، صر اص ، ص1.5.2/
، صر صر اص ، ص/Product_1/FireOS/ (VISITED)
، صر صر اص ، صReadMe.txt
، صر صر صٌ ، صcatscratch.tar.gz
، صر صٌ ، ص2.0.0/
، صر اص ، ص/Product_1/FireOS/ (VISITED)
، صر اص ، صReadMe.txt
، صر صٌ ، صcatscratch.tar.gz
، اص ، صLinuxARM/
، صر اص ، ص/Product_1/ (VISITED)
، صر اص ، ص2.0.2/
، صر صر اص ، ص/Product_1/LinuxARM/ (VISITED)
، صر صر اص ، ص3rdpartytools/
، صر صر صر اص ، ص/Product_1/LinuxARM/2.0.2/ (VISITED)
، صر صر صر صٌ ، صcross_compiler_1.1.0.tar.gz
، صر صر اص ، صReadMe.txt
، صر صر صٌ ، صbloodyscratch.tar.gz
ںصاص ںس3.0.1/
```

```
، صر صر اصر ، ر/Product_1/LinuxARM/ (VISITED)
، صر صر اصر ، ر3rdpartytools/
، صر صر صر اصر ، ر/Product_1/LinuxARM/3.0.1/ (VISITED)
، صر صر صر صرٓ ، رcross_compiler_2.0.0.tar.gz
، صر صر اصر ، رReadMe.txt
، صر صر صرٓ ، رbloodyscratch.tar.gz
، صر صرٓ ، ر4.0.0/
، صر اصر ، ر/Product_1/LinuxARM/ (VISITED)
، صر اصر ، ر3rdpartytools/
، صر صر اصر ، ر/Product_1/LinuxARM/4.0.0/ (VISITED)
، صر صر صرٓ ، رcross_compiler_5.0.0.tar.gz
، صر اصر ، رReadMe.txt
، صر صرٓ ، رbloodyscratch.tar.gz
، صرٓ ، رiOS/
، اصر ، ر/Product_1/ (VISITED)
، اصر ، ر1.1.1/
، صر اصر ، ر/Product_1/iOS/ (VISITED)
، صر اصر ، رReadMe.txt
، صر صرٓ ، رitchyscratch.tar.gz
، اصر ، ر1.2.0/
، صر اصر ، ر/Product_1/iOS/ (VISITED)
، صر اصر ، رReadMe.txt
، صر صرٓ ، رitchyscratch.tar.gz
، صرٓ ، ر2.0.3/
، اصر ، ر/Product_1/iOS/ (VISITED)
، اصر ، رReadMe.txt
، صرٓ ، رitchyscratch.tar.gz
| ، رlogin.html
ٓ ںrenamed_index.html

Statistics: 41 files, 50 directories, 25 duplicated (dirs), 0 unreadable (files)
Time Elapsed: 1.050159 secs
Success: process exited with code 0.
```

Within the CLI parameters, Basic Authentication is supported. If choosing the -testfiles option to determine whether a file has will yield 403, such files are seen in the tree like:

```
...
، صرٓ ، ر*[itchyscratch.tar.gz]**
| ، رlogin.html
ٓ ںrenamed_index.html
```

Where itchyscratch.tar.gz is the unreadable file that provides a HEAD request status code of 403.

Out of Time:

Knowing I have to take a long break from contributing to WebTree, I decided to post WebTree. I expect to find some issues or desires to improve certain areas with the following:

1. The Xpath being used is simple. A more complex path would be an improvement.
2. Output to a PlantUML tree as an option.
3. Whole, rather than skeletal, files could be saved.
4. This example program should be converted into a package.
5. More testing of all CLI parameters (on varying platforms such as Linux and Windows)

Note: The above means that this program a work in progress.