DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout 22—April 13, 1995

Notes on Design Project #1—WWW Naming

This note records various observations about the first 6.033 design project, made by Jerry Saltzer in the course of reading the project papers of his two recitation sections. Many of the ideas in these observations were contributed by students. Unique ideas, suggested by only one or two people, are identified with their source. Of course people in other recitation sections probably also came up with some of the same ideas, so the attributions here should not be considered exhaustive.

I. Desiderata

The design project handout suggested four desiderata for a successful solution. Here are some additional desiderata (not all of the desiderata can be satisfied simultaneously, of course):

a. It should not be necessary to modify existing Web pages.

b. It should not be necessary to modify existing Web browsers.

c. It should not be necessary to modify existing Web servers.

d. Any changes to web protocols, servers, or browsers should be backwards compatible. That is, a new server or browser should interoperate with old servers, browsers, and web pages without creating new problems.

e. It should be possible to move a hierarchy of web pages filled with relative links to a different server [Eric Boyd]. (This desideratum may be identical to the combination of the two desiderata that one can rename or move a hierarchy within a server and also that different servers can AFS-mount and export the same hierarchy of web pages.)

f. It should be possible to move just the home page for a group of related pages anywhere, including to another server, without having to modify its contents [Mike Montwill].

g(1). Interpretation of ".." in relative names should uniformly be semantic, as called for by UNIX.

g(2). Interpretation of ".." in relative names should uniformly be syntactic, as called for by the Web URI specification.

g(3) The semantic and syntactic interpretations of ".." should always come out the same, so it doesn't matter which is used.

h. URL's should have bounded length [Philip Lisiecki, Mark Neri].

i. If a URL goes through a UNIX symbolic link, to the extent possible all relative URL's built from it should also go through the link, for consistency [Philip Lisiecki].

II. Additional problems not brought up in the project handout

a. If one of the BAD Web link references were via an NFS mount of the 6.033/www directory, then nothing could fix that bad link! [Jean-Emile Elien]

b. Some solutions may interfere with some server implementations of access control. (Since we have not discussed access control yet, students are not expected to notice or discuss this aspect. In addition, some Web implementations of access control are so arcane that it is not clear how to avoid interfering with them.)

c. A so-called BAD Web link could set things up so that a later Web link will accidentally lead to some file that happens to have the same (local) name as the one intended. The client would then retrieve something the web-page author didn't expect [Mike Montwill, Lewis Girod]. Worse, if there is another file lying around the server that has the same name as the one intended, there is a good chance that it is a slightly different version of the intended file, so the mistake may go unnoticed and be the source of great confusion. This problem can also be turned into a feature, in which the page you get for a particular Web link depends on which Web link you followed to get to the containing page. [Daniel Lee.]

III. Technical description of the problem, in 6.033 terms

The Web browser supplies an implicit closure for relative names (also called "partial URL's") found in Web pages. The implicit closure it supplies is simply the URL that the browser used to retrieve the page that contained the relative name, truncated back to the last slash character. This closure is the name of a directory at the server that should be used to resolve the (first component of) the relative name.

Some servers provide a URL namespace by simply using the local (for example, UNIX) file system namespace. When the local file system namespace allows synonyms (symbolic links and NFS mounts are two examples) for directory names, the mapping of local file system namespace to URL namespace is not unique: There can thus be several different URL's with different path names for the same object. Trouble can arise when the object that has multiple URL's is a directory whose name is used as a closure.

The specific problem can arise when interpreting the relative name "..". This name is, conventionally, the name for the parent directory of the current directory. UNIX provides a semantic interpretation: look up the name ".." in the current directory, where it evaluates (in inode namespace) to the parent directory. The Web, in contrast, specifies that ".." is not a name to be looked up in some context, but rather a syntactic signal to modify the implicit closure by discarding the least significant component of the directory name. Despite these drastically different interpretations of "..", the result is usually the same, because the parent of an object is usually the thing named by the next-earlier component of that object's path name. The exception (and the problem) arises when the syntactic modification is applied to a URL that contains a synonym for a directory name. If the path name of the synonym does not come through the directory's parent, syntactic interpretation provides an implicit closure different from the one that would be supplied by semantic interpretation.

The problem can be fixed in at least three fundamentally different ways:

1. Arrange things so that the current implicit closure always works.
2. Do a better of job of choosing an implicit closure.
3. Provide an explicit closure.

IV. A misleading characterization of the problem

One might suggest that the implementor of the server (or the writer of the pages containing the relative links) failed to take heed of the warning in the Web URL specifications for path names that "The similarity to unix and other disk operating system filename conventions should be taken as purely coincidental, and should not be taken to indicate that URIs should be interpreted as file names." (Tim Berners-Lee, Universal Resource Identifiers: Recommendations.) That suggestion, however, is misleading.

Unfortunately, the problem is built in to the Web naming specifications. Those specifications require that ".." be interpreted syntactically, yet they do not require that every object have a unique URL. Unambiguous syntactic interpretation of relative names that begin with ".." requires that the closure consist of a unique path name. Since the browser derives the closure from the path name of the object that contained the relative name, and that object's path name does not have to be unique, it follows that syntactic interpretation of relative names that begin with ".." will intrinsically be ambiguous. When servers try to map URL path names to UNIX path names, which are not unique, they are better characterized as exposing, rather than causing, the problem.

That analysis suggests that one way to conquer the problem is to change the way in which the browser acquires the closure. If the browser could somehow obtain a canonical path name for the closure, the same canonical path name that the UNIX system uses to reach the directory from the root, the problem would vanish.

V. Solutions

There are many prospective solutions, each with advantages (+) and disadvantages (-). Note that solutions 1.d/e/f, 2.a/b/c/d, 3.c, and 4.e/h fully satisfy the four original desiderata.

1. Make the current implicit closure work, by imposing a discipline on web page maintainers.
   + no programming required; preserves HTML, HTTP, server, and client implementations.

   a. Require that all relative filenames refer to points hierarchically below the URL entry point, and place all the web pages there. For the case of the 6.033 handouts directory, move the html directory into the www hierarchy and revise the relative pathnames in web links.
      - Constrains allowable organizations. The html directory belongs equally well as a child of handouts and a child of www.
      - Have to modify existing web pages to comply, though this could be done accurately and mechanically with a script.
      - Hard to enforce--in a multi-user system, someone unconnected with www management may construct a UNIX symbolic link into the middle of the hierarchy of web pages [Philip Lisiecki].

   b. Same idea as 1.a. In addition, forbid use of .. in relative filenames.
      - Constrains allowable organizations. The html directory belongs equally well as a child of handouts and a child of www.
      - have to modify existing web pages to comply, though this could be done accurately and mechanically with a script.

   c. Require that all relative filenames refer to points hierarchically below the URL entry point, and install soft links to make it appear that all web pages are located there. In addition, forbid use of .. in relative filenames. For the case of the 6.033 handouts directory, install the UNIX symbolic link "html --> /afs/athena/course/6/6.033/handouts/html" in the 6.033/www directory, and excise the "../" from all relative pathnames that lead to handouts.
      - If the locker moves to a different location, the UNIX soft links must be adjusted.
      - have to modify existing web pages to comply, though this could be done accurately and mechanically with a script.

   d. Move the burden of interpreting ".." from the Web software over to UNIX by taking the ".."'s out of the web links and slipping them into strategically placed UNIX symbolic links instead. For the example of the 6.033 locker, one would place a UNIX symbolic link "html --> ../html" in the 6.033/www directory, and excise the "../" from all relative pathnames that lead to handouts (Chris

Shabsin).

+ meets all the original desiderata.

- have to install soft links and modify existing web pages to comply, though this could be done accurately and mechanically with a script.

e. same as c. or d., but instead of installing a UNIX soft link to the directory containing other web files, install a separate UNIX soft link to each individual file to which a relative web link exists.

+- same trade-offs as c. and d.

- lightly more effort to maintain by hand

+ easier to automate maintenance.

f. Move the burden of interpreting ".." from the Web software over to UNIX by installing a symbolic link named ".up" that points to the parent directory in each UNIX directory that holds a web page, and replacing all ".." constructions in relative links with ".up" instead. (Eric Nygren and Steve Niemczyk.)

+ meets all the original desiderata.

- have to install soft links and modify existing web pages to comply, though this could be done accurately and mechanically with a script.

2. Better implicit closure. Let someone (UNIX) do it who knows how to provide the correct Implicit closure. Modify both the server and the client to not do any syntactic manipulation of relative filenames. Instead, pass both the closure and the relative file name along to the server's underlying file system, which will look up ".." in the directory named by the closure.

+ meets all the original desiderata

- but all specific implementations have problems.

- compromises some servers' attempts at security based on pathnames rather than on the underlying UNIX file access control system. To preserve security, the server would have to use the UNIX file system primitives to convert the relative filename into an absolute pathname, then invoke the http access control mechanism on this absolute pathname, and then actually open the file for transmission to the client.

a. Client concatenates the current implicit closure and the relative file name, and sends the resulting URL (containing unprocessed ".."s) to the server. The server passes the path name in the URL along to its underlying file system.

- Unbounded-length URL's. The client now does not have a compact URL for the retrieved page; if the retrieved page contains another relative path name, the client will have to repeat the surgery on the previously altered URL; each such followed link produces a longer URL.

b. Same idea as 2.a. In addition, the server constructs a valid URL for the retrieved page with the help of link-following features of UNIX and sends that back to the client along with the retrieved page.

- Unnecessary mechanism. If the server can send with every retrieved page a valid URL for that page, then it can with only slightly more work send a valid URL that contains an absolute pathname (that is, that doesn't go through a UNIX soft link) and the client can resort to syntactic interpretation of the "..".

- The client needs an equivalent way to construct valid URL's when it loads files from a local file system.

c. Maintain symlinks when possible. Same idea as b., but the server converts a path name through a symbolic link to an absolute path name only if a .. relative reference requires it.

+- same trade-offs as b.

+ If the user squirrels away the resulting URL it will survive movement of the page hierarchy to a different location on the server, assuming that the server's symbolic link changes to point to the new location.

d. "Remote reference": Same idea as 2.b, except that the client sends its implicit closure together with the relative reference as two distinct objects to the server for resolution. The server always sends back a valid URL for the resulting page, for use in following relative links found in that page [Jean-Emile Elien].
 +- same trade-offs as b.
 + Server can decide to implement either semantic or syntactic interpretation of "..", as it prefers.

3. Explicit closures.

a. Use a BASE tag containing the URL of the base www directory at the beginning of every page, and base all relative names on that address. (this is actually a discipline on the web page maintainers.)
 + The BAD links no longer lead to weird results.
 - Forces use of the server named in the BASE tag on every reference
 - Forces use of the server even when reference is completely local.
 - If locker moves to a different location, every BASE tag must be changed.
 - If the server host name changes, every BASE tag must be changed.

b. Modify the client interpretation of the BASE tag to be relative to the host on which the containing URL was found. Use a BASE tag which contains just an absolute path name of the base www directory at the beginning of every page.
 + The BAD links no longer lead to weird results.
 - If locker moves to a different location, every BASE tag must be changed.
 - Works only if every server uses the same absolute path name for the referring page (Jonathan Sheena, Chris Shabsin)

c. Modify the http server to always construct a canonical URL for any page requested, and include that URL in the URI (or Location) field of the http response header. This canonical URL would contain the absolute path name of the page, obtained by locating the page with the originally requested URL and then following up the ".." chain backwards to the root. The client, in turn, would replace the URL it originally requested with the one returned by the server, as the basis for the closure used to interpret relative path names. (It is said that some browsers already do this replacement when they see a URI in the http response header [Mark Neri].) The canonical URL for a page should produce the expected result under syntactic interpretation of all relative names found in the page. If loading a local file, the browser would construct the absolute pathname of the page.
 + meets all original desiderata.

d. Modify the http server to always construct a canonical URL as in solution c, and send that URL to the client in the BASE tag of the page [Jamie Coffin].
 - Abstraction boundary violation. http server shouldn't be messing with file contents.
 + meets all original desiderata.

e. Place a BASE element in a separate file that has a well-known name in the same directory as a group of web pages, with the notion that it provides a context for relative path names of all the web pages in that directory. The client, when it encounters a relative path name, can construct a URL for and retrieve the file containing the BASE element using http, as a separate transaction [Nimisha Mehta].
 - Forces use of the server named in the BASE tag on every relative reference.
 - If the hierarchy is moved, the well-known file in each directory must be modified.

4. Other ideas, mostly introduced to be shot down.

a. Along with each page, send a copy of the entire directory tree from the client to the server [Jean-Emile Elien, Daniel Lee]

- expensive

b. forbid relative path names. Use full URL's for all web links.
- Can't move groups of related files without doing a lot of internal rework.
- Can't use alternate servers, because URL's specify the server name.
- Local file loading leads to use of the server when following web links.

c. forbid use of UNIX links. enforce by switching server follow-link feature OFF.
- Constrains organization of information on server, especially when some materials are accessible both via the web and in other ways.

d. server examines all outgoing html files, expands relative links to be full URL's with proper absolute path names before sending [Jamie Coffin, Jonathan Sheena].
- Abstraction boundary violation. http server shouldn't be messing with file contents.

e. Add a resolve server. If a request for a page that involved a relative name fails, then send a remote reference request (as in solution 2d) to the resolve server. The server sends back a correct URL, if one exists, and the client then requests that page [Doug Wyatt].
+ meets all desiderata. (if local use of resolve server also works.)
- depends on the first inquiry failing by not finding a page, rather than failing by retrieving the wrong page.

f. Add state to the server. Have it remember a working directory for each client [Gregory Pal, Terry Chau].
- server has no way to know when to discard state.

g. Replace all URL's in the WWW page hierarchy with URN's. Provide a name service that knows about alternative servers. Upgrade the client to automatically try different alternative servers. Also upgrade the client to notice that a URN is local and look directly for the file.
- Requires that there be a registered URN for every page. It would be a hassle to add pages.

h. Implement BASE references with URN's. Provide a name service that knows about alternative servers. Upgrade the client to automatically try different alternative servers. Also upgrade the client to notice that a URN is local and look directly for the file. Finally, change the client to resolve a BASE URN to a URL before interpreting relative names (Eric Mumpower).
+ could meet all desiderata--assuming a lot of implementation details are done right.

i. Just say no. Stop using the web. [Rachel Caileff]
+ enormous amounts of wasted time could be put to more productive use.

Bibliography

The Bibliography on the ENCOUNTER page has been replaced with a link to an HTML version of the following bibliography. As with the solutions, many of these references were pointed out by students who did the project.

General references about the World-Wide Web

Tim Berners-Lee.
The World-Wide Web.
Home page of the World-Wide Web project office, undated.
http://www.w3.org/hypertext/WWW/TheProject

Tim Berners-Lee.
World-Wide Web Summary.
World-Wide Web project office, undated.
http://www.w3.org/hypertext/WWW/Summary

Tim Berners-Lee, et al.
WWW Bibliography.
World-Wide Web project office, undated.
http://www.w3.org/hypertext/WWW/Bibliography.html

HyperText Markup Language (HTML)

Daniel Connolly.
HyperText Markup Language (HTML).
World-Wide Web project office, June, 1993.
http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

Mark Andreessen.
HTML Primer.
http://www.ncsa.uiuc.edu/demoweb/html-primer.html

Ian Graham
HTML Documentation.
On-line version of *The HTML Sourcebook.* John Wiley and Sons, ISBN 0-471-11849-4, March 1995.
http://www.utirc.utoronto.ca/HTMLdocs/NewHTML/intro.html

Daniel Connolly.
HTML Design Notebook.
World-Wide Web project office, January 24, 1995.
http://www.w3.org/hypertext/WWW/People/Connolly/drafts/html-design.html

(Tim Berners-Lee.)
HTML 1.0 Specification (Obsolete).
Internet Engineering Task Force Draft, undated.
http://www.w3.org/hypertext/WWW/MarkUp/HTML.html

Daniel Connolly.
HTML 2.0 Specification.
World-Wide Web project office, March 29, 1995.
http://www.w3.org/hypertext/WWW/MarkUp/html-spec/html-pubtext.html"

Earl Hood.
HTML 2.0 DTD (Document Type Definition).
World-Wide Web project office, November, 1994.
http://www.oac.uci.edu/indiv/ehood/html2.0/DTD-HOME.html

Dave Raggett.
HyperText Markup Language Specification Version 3.0 (Formerly HTML+).
Internet Engineering Task Force Draft , undated.
http://www.hpl.hp.co.uk/people/dsr/html/CoverPage.html

Dave Raggett.
HTML 3.0 DTD (Document Type Definition).
Apparently unofficial, March 24, 1995.
http://www.hpl.hp.co.uk/people/dsr/html/html3.dtd

HyperText Transfer Protocol (HTTP)

(Tim Berners-Lee.)
Basic HTTP .
Internet Engineering Task Force Draft, undated. (and obsolete)
http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html

Roy T. Fielding.
Hypertext Transfer Protocol (HTTP).
Internet Engineering Task Force Working Group paper, undated
http://www.ics.uci.edu/pub/ietf/http/

Tim Berners-Lee, Roy T. Fielding, and H. Frystyk Nielsen.
Hypertext Transfer Protocol -- HTTP/1.0
Internet Engineering Task Force Draft, March 8, 1995.
http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-v10-spec-00.txt

World-Wide Web Linking (UR*'s)

Tim Berners-Lee.
WWW Names, Addresses, URI's, URL's, and URN's.
World-Wide Web project office, November, 1994.
http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

Tim Berners-Lee.
Uniform Resource Locators.
Internet Engineering Task Force, URI Working Group, March 21, 1994.
http://www.w3.org/hypertext/WWW/Addressing/URL/Overview.html

Tim Berners-Lee.
Uniform Resource Identifiers (RFC 1630).
Internet Engineering Task Force, URI Working Group, March 21, 1994.
http://www.w3.org/hypertext/WWW/Addressing/URL/URI_Overview.html

(Tim Berners-Lee.)
Partial (relative) form .
World-Wide Web project office, November, 1994.
http://www.w3.org/hypertext/WWW/Addressing/URL/4_3_Partial.html