This is the accepted version of a paper published in . This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Magnusson, L. (1996)
Local News: Exploring Web-based Threaded Discussion Server Functionality
Sys Admin Magazine - The Journal for UNIX Systems Administrators, 5(10)

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-77949
Local News - Exploring Web-based Threaded Discussion Server Functionality

Lars Magnusson

Although I installed my first Web server in 1994, I didn't pay much attention to Web technology itself until last autumn when I was at a convention and saw a demonstration by Swedish Telecom, Telia. They showed an application for their Net documentation - a text repository able to check in new or changed texts. It was not as streamlined as Tim Berners-Lee's original Web system, but it was quite functional.

Unfortunately, Telia wasn't interested in releasing the CGI script they used, partly because they felt it was not ready. But it inspired a lot of the conference participants and gave me an excuse to take a closer look at what Web applications really could do. There was a lot of discussion about text repositories going on at my primary employer, a municipal council, so it was natural to look into the possibility of replicating Telia's application.

Since I did not have access to a Web server with CGI capability at that time, I tried using the posting capability instead, and I found that making a user friendly repository with that method was not possible. This was mainly because the present HTML standard does not support a predefined Subject-line in a post, which is necessary to get reference information into the return post.

I was drawn into a discussion about FirstClass, a Canadian conference system that used a proprietary protocol. Naturally, the idea arose to look at a local Web-based conference server, with an interface not unlike Netscape 1.x's Netnews. (See Listing 1, lnews.sh.)

After exploring FirstClass, I found I now had some code, including an old mailserver (see "An Internet Gate for a Local Email System," Sys Admin Jan/Feb 1995: 49-56), that could be used. And, with the post option in Web forms, I gained much of the control over the flow that I previously lacked. I was then faced with five general problems to solve; I needed to:

1. Restore the mailed form to a structured text, handling both older Netscape 1.x string data as well as newer 2.x MIME-code data.

2. Transform the HTML/MIME codes for special chars, especially the Swedish chars (most of which are vowels-very important in getting words correct).
3. Generate a message-id in form of a index number

4. Build the resulting HTML page

5. Create a theaded index page, similar to that of Netscape Netnews

Restoring Mail and Transforming Special Characters

Submissions sent via the Web-forms interface come in as conventional email, potentially manipulated by several intervening mail transport agents. To restore the email back to a formatted letter, I needed to resolve other problems:

1. Netscape version 1.x and older delivers form data in a continuous string, whether it is through CGI or a post.

2. Both rmail and mailx/Mail break lines at 80 characters.

3. Netscape 2.x delivers some posted data in a MIME-format, so I had to decide how to identify the type by looking in the mailheader for the MIME header.

Pre-Netscape 2.x

To allow use of versions of Netscape prior to 2.x, I was faced with both dividing the string at certain points and joining lines elsewhere. Most UNIX tools do not handle strings longer than 1024 characters, so I wanted to keep lines as short as possible. In examining this problem, I also found that the HTML code for CR/NL, %0D%0A often got broken by the 80-character border.

So, I started breaking the lines that could be broken, and joining the lines that should be joined. Then, I had to break "joined" lines. Only then could I substitute the special characters. The following segment of code accomplished the required task:

```bash
sed 's/%0D%0A/\n/g
s/\n/\
/g' infile nawk '{
if (index($0,"\n") != 0) {
oldlin=substr($0,1,length($0)-2)
getline newlin
print oldlin"newlin
} else {
print $0
}
}' sed 's/%0D%0A/\n/g

....... # rest of substitutions follows
```

After experimenting, I used awk to join the lines, as shown above, and used sed for the substitutions. I could have used a single awk script, but I got better performance by piping it through one sed script, then through awk, and finally to the last sed script.

Post-Netscape 2.x
Having a MIME-encoded reply from a user with Netscape 2.x simplified some of the process, because the filter for special characters was much less complicated. But I still had to extract the subject-line, the letter, and the message-id. By postulating that the replies layout was controlled by the form and not by the viewer, I extracted the line numbers of the MIME field-headers. These were then used to make three sed scripts that cut up the reply, stored the subject and message-id in their respective shellvars, and stored the main text in a temporary file.

After all this, I finally had a readable letter - text that was easy to process further. I then retrieved some additional information from the reply. These values were also stored in shell variables for future use. Since the mail address handling seems to be different for different viewers, I paid some extra attention, parsing mail addresses and usernames from those standard formats I know of.

**Generation of the Message-id**

In Netnews, relationships between articles are maintained by adding a special header line, with the parent article's message-id written into it. If it's a new thread, then the header line is empty. It is simple to make this message-id, but it demands large resources when doing the threaded hierarchy later.

I wanted a simpler, less demanding solution, allowing rapid threading of hundreds of articles. I therefore chose to connect the message-id to the filename, so they reflected each other and thus simplified threading.

I chose to create a hierarchical numbering, in which a first post had a single number, say, 7. Then replies would have numbers 7.1, 7.2, replies to replies 7.2.1 and 7.2.2 and so forth. This gave me the same well-defined structure as an outline.

To separate the groups, I gave every group a beginning character from A to Z. It limited the present version to 26 groups, but that can easily be changed by using double letters or some other pattern. By combining the group char and the number and adding a suffix [.html], I had a filename I could use. However, since I started the design under MS-DOS with MKS Toolkit, I was faced with the limitations of MS-DOS filenames (e.g., only one dot allowed in the filename). I substituted all but the suffix dot with a underscore, so in reality the filenames look like B7_2_3.html instead of B7.2.3.html, while the message-id still says B7.2.3. This is only a minor inconvenience though. The MS-DOS limit of eight characters in the leading portion of the filename also limits the number of postings that the system is capable of handling.

Note that msg-id is the message-id for the parent posting. When a discussion participant posts a new thread or a reply, the message-id for the new posting is unknown. lnews has to calculate it when the post arrives by adding one to the last discussion thread number in the appropriate group level. With help of find, I retrieve the last valid post in the actual level by extracting the parent posting.
message-id from the reply and using it as value for the search. The find action looks like this:

```
find dir( -name "msg-id.html" -o -name "msg-id_*.html" -a ! -name "msg-id_*_*_.html") -print tail -1
```

If this is a new thread, the parent-id is always 0, forcing a special find search to return the last main level number. Otherwise, the program looks for the parent-id and for already processed replies in the main level, but not the underlying levels. The ! -name directive negates those subsequent levels searches, so they don't intervene. Then it reads the last name found, which will be used to calculate the new message-id.

Choosing Perl for this function would have simplified the calculation and string handling. But, I wanted to try standard UNIX functions, so I used find, awk, and expr.

**Building the Resulting HTML Page**

I wanted to find a style for the HTML page that was as simple and easy-to-read as possible. I dislike not seeing either top or bottom on long Web pages, so I used the Web form, which allowed having the header and footer visible at the same time. The text was in a 14-line scrollable area in the middle of the page. The buttons in the footer made it easy to skip dull postings, and the small scrollable area was not too uncomfortable.

But, using a form for reading also meant that the user was always in posting mode. I showed this to some colleagues, and they felt that being in reading as well as posting mode at the same time was confusing. So, I separated the text into a Netscape v.1.x Netnews-like reading page and a form-based posting page. The latter was actually the original page, minus some buttons.

I generate the reading page with the suffix .html and the posting page with .xhtml. This means there are two pages for every post, which is both a disadvantage and an advantage. The disadvantage is the increased disk usage - having two pages stored. The advantage is that while the posting form could be generated "on the fly" using CGI, this solution gives a faster response. I have found that most CGI-generators are quite slow. Given present economical disk prices, the additional storage may be more a matter of taste than a real design problem. Java or Inferno might handle this issue more effectively.

**Page Layout**

The design of the pages was done by making a simulation of what I wanted with a Web editor. When I had the correct layout, I took the HTML code for the page and transformed it into print settings in the two awk scripts generating the pages. I also experimented with Netnews-buttons in Netscape, but because the newsreader is a proprietary addition to Netscape, I decided not to use them. Instead,
I created some submit buttons in a form, with the actions I wanted written on them. I then took a screenshot, cutting out the different buttons to JPEG-files. I have had some problems with alignment in the form page and still need to clean up the jpegs a bit, but otherwise this method worked.

Lastly, I had to import some variables, among those, the shell variables I saved in the beginning. Because I originally had some problems transferring shell variables into awk when using MKS Toolkit, I saved these in a temporary data file. This file is read by the subsequent awk scripts, assigning data to the appropriate variables in the scripts. For portability, I have retained this construction. The variables are the poster's mail address and name, the subject, the calculated message-id for the post, the http-tag for the conference main directory (used in the button HREFs), and the mailing address for the conference maintainer.

I used the http-tag because I felt it would be too much work to change the value in every tag in the lnews script when porting the script. By using shell variables, the value needs to be changed at one place in the script. Site-specific data will always be assigned at only one place.

**Make a Threaded Index**

After using tin to read news for some years, I am very fond of getting related postings grouped together. When I saw how the Netscape designers solved this for their newsreader in version 1.x, I felt it was a natural evolution from tin. Although tin's paged approach is neater, with the command menu always available at the bottom, the Netscape index gave a good overview.

With the numbering scheme discussed earlier, I had a lot of the hierarchical structure already in place. But I still needed the program to sort the major numbers in reverse (the highest on top), so the newest post would be seen first as in Netscape. But, a reverse sort would put the replies before the first post in the thread, and I didn't want that. The solution proved to be simple enough as shown in the following code:

```bash
find "/$WEBDIR/$GRP" -type f -name "*.html" -print
nawk 'BEGIN{ FS="/" }{print substr($NF,2)}'
sort +0nr
nawk '{print "'$WEBDIR'/'$GRP'/'$GRP'"$0 }'
xargs egrep "From: Subject: "
nawk '{ .....}
```

First, the program lists the .html files in the directory where the particular conference resides. It then filters out the path and the conference character. Then, the list is sorted in reverse for the first position and sorted normally for the following positions. The full path is then reconstructed. When the sorted list is in order, the program retrieves the poster's name and the subject from the post, and makes an index line that is concatenated to the new index page. A header and a footer is then added to the page. This index page is reconstructed for every new post to the particular conference.
Administration and Problems

For now, the script runs as a mail filter, activated every time a post is delivered to the mailbox. The alias file contains the line: lnews: "/usr/local/bin/lnew". I have also tried to run this under cron, but I prefer the mail filter approach. However, if there are many posts, the cron solution might ease the system load somewhat and allow queueing, as well.

Note that with some mailers the process of checking the mailbox with cron leaves some status letters to the script owner. Therefore, run the script under another UID, so these messages don't foul up the script. Also remember to remove the mailbox for that UID, so you don't run out of disk space, since you need to activate the script every minute.

Site-specific information is stored in the beginning of the shell script, separated into site dependencies and OS dependencies. Some changes may be required in these sections to reflect the platform the script is running on. And, a word of caution: the find code in the message-id generation was created under Solaris 2.x and has only been tested under Solaris 2.x and MKS. The same goes for the threading function in the index generation. The sort +0nr used is running as described under Solaris, but not under MKS Toolkit or Linux. By including the conference letter and having a small awk script doing a ls of each main thread, it is possible to mimic the Solaris behavior. An example will be found at my Web page http://www.abc.se/-m8827/prog.htm.

Security

I use a mail filter, doing active handling of headers and text only with basic UNIX tools. I have never heard about anyone able to break them in this context. The main risk command, expr, is never in direct contact with user input. Therefore, most of those risks associated with CGI scripts and Internet should be neutralized. I would be happy to hear from readers if they see security holes in this approach.

Conclusion

When I started the project, my objective was to test the limits of Web scripts. That exploration showed that there is much to be done by way of adding core functionality to Web scripting. Although the resulting software is functional, with mail filtering I cannot deliver specific information to a specific user, as with CGI. But much of the processing done at a Web site can probably be done this way. For example, a company or organization low on funds could rent some Web pages at a Webhotel, putting up forms that direct the mail back to a small Linux box at the office. A mail filter like this would then take appropriate action, maybe even mailing back requested information. It could also be on a dial-up connection, serving customers/members just like a fancy, direct-connected Web server.
About the Author

Lars has a B.Sc. in Geology from Gothenburg University in Sweden (1980) and a M.Sc. in Petroleum Exploration from Chalmers University of Technology, also in Sweden (1984). He has worked with mining geology/geophysics in Sweden and Greenland, but since the mid-1980s has worked as sysadmin, system manager, and teacher in Sweden, Greenland, and Denmark. At present, he is working as consultant within the areas of UNIX, Internet, and email.