$2.50

MUUGlines

May 1996

Volume 8, Number 7

This Month's Meeting

The Oracle WebServer™ combines the simplicity of the World Wide Wide with Oracle's leading edge database technology to enable a whole new generation of powerful Web applications. During this presentation we will demonstrate an Oracle Web application running over the real internet.

This meeting is scheduled for Tuesday, May 14, at 7:30 PM. Once again, the meeting will be held in the auditorium of the St-Boniface Hospital Research Centre, just south of the hospital itself, at 351 Taché. You don't have to sign in at the security desk — just say you're attending the meeting of the Manitoba UNIX User Group. The auditorium is on the main floor, and is easily found from the entrance.

Editor's Forum

Being the editor, I get to determine the placement of articles in this newsletter. That being the case, I thought this month I'd highlight my desire – right on the front page! – for submissions of new, interesting or bizarre (your choice!) articles for publication in MUUGlines.

I feel that a large part of the shape of a group of this kind is reflected and formed by its newsletter. So here's an opportunity to make this the kind of user group you want.

Should we be more technical? Perhaps you could check out the monthly SIG meeting. Do you want the meetings to be better oriented toward the needs of your business? Inform the board (at board@muug.mb.ca). And send in those articles!

As a bonus, this is your chance to be a Published Author!

Form & Function - Using Forms On the Web

Part 1 - Coding Forms In HTML

By Gilbert Detillieux, Computer Science, University of Manitoba

Internet fever has reached an all-time high. The World Wide Web had been not only one of the biggest catalysts in popularising the Internet, but promises to be one of core technologies for individuals and businesses who are adopting the Internet as a communications medium. The easy-to-use interface of modern web browsers, and the attractive, multi-media content of web-based hypertext, make the web a good way of providing information for the masses.

However, this hypertext interface on its own has its limitations. It is still essentially a one-way communications medium, with producers on one side and consumers on the other. To be an effective business communications tool, there's a need for interaction. There's also a need to provide dynamically obtained, up-to-date information, rather than just static pages of hypertext. Fortunately, the web has evolved to meet these needs, with extensions such as fill-out forms, server-side processing, and most recently, client-side programming capabilities.

In this article, we'll look at how fill-out forms are done in HTML. Part 2 will look at the Common Gateway Interface (CGI), which lets you set up scripts to do server-side processing. Finally, Part 3 will look at Javascript, a new scripting language developed by Netscape, to do client-side programming, and processing of forms.

A Sample Form

Let's suppose we are managing a mailing list for the membership of some organisation. We now want to automate the task somewhat, by allowing applications, renewals, and address changes to be handled via the web. We could set up an electronic form to allow users to submit the data, then use a program of some sort on the web server that would process the data, and manage some sort of database. We'll start with the basics.

Continued on page 2

In this issue

HTML Forms, continued from page 1

```
<FORM METHOD="POST" ACTION="URL for CGI program">

... form input fields ...

</FORM>
```

A form in HTML is enclosed within FORM tags. On the opening tag, we specify an ACTION, with is usually the URL to invoke a CGI program on the server, which will process the form input data. We also specify a METHOD of either GET or POST, which indicates the way the form input is to be passed along to the server. (We normally use POST for any forms that will contain a fair amount of data; GET can be used for simple queries, where the amount of data is small.)

Since we'll only get to CGI programs in part 2, we'll use a different type of ACTION for testing our form:

```
<FORM METHOD="POST"
ACTION="mailto:gedetil@cs.umanitoba.ca">
```

Using a "mailto:" URL as an action may not be supported by your browser. Netscape introduced this extension, and others have since followed suit, but many browsers don't support it. However, mailing forms to yourself is a useful tool for testing out your forms.

Now we can put some input fields in the form. The simplest type is a one-line text input field:

```
<b>Name <em>(your full name)</em>:</
b><br>
<INPUT SIZE=40 NAME="fullname">
```

This sets up a text box which can display one 40 character line of text. The user will be able to enter and edit a string of more than that size, but only 40 characters will be visible at any time. The NAME you give is arbitrary, but you can think of it as a variable name for the input field (this analogy will be useful later).

An initial, default string of text can be included with the VALUE attribute on the tag:

```
<INPUT SIZE=40 NAME="fullname"
VALUE="Joe Blow">
```

Multiple lines of text input are also easily handled:

```
<b>Address:</b><br>
<TEXTAREA NAME="address" ROWS=5
COLS=40></TEXTAREA>
```

This sets up a text box consisting of 5 lines of 40 characters. Scroll bars on the box allow you to view additional text, if it doesn't fit within the specified window size. You can also have initial, default text, by enclosing it between the opening and closing TEXTAREA tags.

In addition to text input, we may want to allow the user to select from a set of listed options. There are different ways to accomplish this, such as check boxes, radio buttons, and option menus. Here's an example of radio buttons:

```
<b>Application for:<br></b>
<INPUT NAME="status" TYPE="radio"
VALUE="New" CHECKED>New
membership
<INPUT NAME="status" TYPE="radio"
VALUE="Renewal">Renewal
<INPUT NAME="status" TYPE="radio"
VALUE="Change">Address change<br>
```

Note that all three buttons have the same variable name; this is what makes them work together as a group, where only one can be selected at any given time. The "status" variable will take on the value associated with whichever button is selected when the form is submitted. You can use the CHECKED attribute to indicate the default selection.

Check boxes are similar, but have a TYPE="checkbox" attribute. The VALUE attribute is assigned to the field if the check box is checked. Of course, each check box should have a unique NAME attribute.

For longer lists of options, check boxes and radio buttons may not be appropriate. Option menus can be set up, either as pull-down menus or as scrollable lists, using the SELECT tag. I won't go into detail here, since it's beyond the scope of this article.

Finally, we need a couple special buttons on our form, to make it useful:

```
<INPUT TYPE="submit" VALUE=" Submit
Form ">
<INPUT TYPE="reset" VALUE=" Clear
Form ">
```

The submit button is used to submit the form, i.e. to send it to the URL specified in the form's ACTION attribute. The reset button, which is not really necessary but is a nice touch, will clear all of the form's fields to their initial default state.

So, now we can put it all together, into a complete HTML document:

```
<HTML>
<HEAD>
<TITLE> Membership Form </TITLE>
</HEAD>
<BODY>
<H1> Membership Form </H1>
Please enter all of the relevant information,
then use the Submit button
at the bottom of the page.
<FORM METHOD="POST"
ACTION="mailto:gedetil@cs.umanitoba.ca">
<b>Application for:<br></b>
<INPUT NAME="status" TYPE="radio"
VALUE="New" CHECKED>New
membership
<INPUT NAME="status" TYPE="radio"
VALUE="Renewal">Renewal
<INPUT NAME="status" TYPE="radio"
VALUE="Change">Address change<br>
<p><b>Membership Number <em>(if you're
already a member)</em>:</b><br>
<INPUT SIZE=40 NAME="membnum">
<p><b>Name <em>(your full name)</em>:</
b><br>
<INPUT SIZE=40 NAME="fullname">
<p><b>Address:</b><br>
<TEXTAREA NAME="address" ROWS=5
COLS=40></TEXTAREA>
<p><b>Phone <em>(daytime)</em>:</
b><br>
<INPUT SIZE=20 NAME="phone">
<p><b>E-mail <em>(if applicable)</em>:</
b><br>
<INPUT SIZE=40 NAME="email">
```

```
<p>
<INPUT TYPE="SUBMIT" VALUE="
Submit Form ">
<INPUT TYPE="RESET" VALUE=" Clear
Form ">
</FORM>
</BODY>
</HTML>
```

Form Submission

Now, what happens when we press the submit button? All of the form's input fields are encoded into a special string, and passed along to the specified URL. In the case of our example, where we

Continued on page 4

SIG Meeting

TeX and LaTeX on Linux, presented by Michael Doob, of the University of Manitoba, Math Department. TeX and Linux have similar historiesin many ways. Both are large (suites of) software, both started out as a one-man project that grew with the help of innumerable volunteers, both are available for free, or can be purchased on a CD-ROM for convenience, both are completely obtainable over the Internet, and both have had a major impact on a broad spectrum of users.

TeX is the older of the two projects, and in fact a TeX distribution (the Linux T series) was one of the earlier packages available for use with Linux (it is remarkable in itself that after a dozen years TeX remains the premier typesetting engine for mathematics). Different packages available for use with TeX have been evolving to address some of its deficiencies, and now Linux users are contributing to the TeX world. We'll talk about a relatively new release, teTeX by Thomas Essers, which was designed to run under Linux, but is now spreading to other UNIX platforms.

The MUUG SIG will be on again this month at ISM (400 Ellice Avenue) on May 21, 7:30 PM.

Doom on Linux

by Steve VanDevender <stevev@jcomm.uoregon.edu>

What's DOOM?

DOOM is a wildly popular action game developed by id Software where you, a trained space marine, have to fight your way through a series of increasingly hellish moonbases that have been invaded by demonic creatures. It features stunning realtime texture-mapped 3-D graphics, chilling stereo sound effects, and frenetic gameplay. You can also play DOOM in multi-player cooperative or Deathmatch games using modems, IPX networks, and TCP/IP networks.

id distributes a full-featured shareware version with the 9 levels of DOOM Episode 1, "Knee-deep in the Dead", and for a $40 registration fee you can buy two more episodes, "The Shores of Hell" and "Inferno". A commercially distributed sequel, DOOM II: Hell on Earth, is being sold in stores. DOOM enthusiasts have decoded the formats of DOOM resource files with the blessing and support of id's programmers and created literally hundreds of user-designed levels (often called WADs or PWADs) to supplement the 27 levels in the registered version of DOOM or the 32 levels in DOOM II (you must have the registered version of DOOM or DOOM II to use these add-on levels).

Although primarily marketed for MS-DOS, DOOM was originally developed on NeXT systems and has been or will be ported to SGI systems, Macintoshes, the Atari Jaguar, MS-Windows, and more. Now, a Linux port done by David Taylor of id is available that runs with Linux and the Linux SVGA library or XFree86.

This document

I have compiled this document from my own experiences and information other people have posted on the net. Whenever I remembered to, I have credited sources of information. If I missed crediting you, I'm sorry. This document is meant to provide information to help you install and use the Linux ports of DOOM. It does not attempt to answer questions about Linux or DOOM that are better answered by other sources, like how to recompile your Linux kernel or how to play DOOM. References to some of these other sources are included.

Please feel free to send comments, changes, and additions to me <stevev@jcomm.uoregon.edu>. If I use them, I will try to credit you. Since things change rapidly, sometimes pieces of this document will be out-of-date or just plain wrong. When I am looking for more information on a subject or have a parenthetical comment, I will enclose it [in brackets].

Where to get Linux DOOM and DOOM add-ons

David Taylor has uploaded his releases of Linux DOOM to sunsite.unc.edu, directory /pub/Linux/games/doom . linuxxdoom.tgz contains the X executable, while linuxsdoom.tgz contains the SVGAlib executable. Each is a gzipped tar archive of the executables and a README with documentation for the game. doom1v18.tgz is the shareware DOOM data file doom1.wad. If you want to get a complete kit all in one file, retrieve linux-doom-1.8.tar.gz , which contains:

HTML Forms, continued from page 3

used the POST method, and mailed the results to ourselves, we would receive a message something like the following:

```
From: "Gilbert E. Detillieux"
<gedetil@cs.umanitoba.ca>
To: gedetil@cs.umanitoba.ca
Subject: Form posted from Mozilla
Date: Mon, 25 Mar 1996 10:39:15 -0600
X-Mailer: Mozilla 2.0 (X11; I; SunOS 4.1.3
sun4m)
X-Url: http://www.cs.umanitoba.ca/~gedetil/
form/form.html
Mime-Version: 1.0
Content-Type: application/x-www-form-
urlencoded
Content-Length: 161

status=New&membnum=&fullname=Gilbert
+Detillieux&address=123+Mulberry+Lane%0D
%0AWinnipeg%2C+MB%0D%0AR3R+3R3
&phone=%28204%29+5551212&email=gedetil
@cs.umanitoba.ca
```

The message body above is actually sent as one continuous line. The MIME-format headers tell us what to expect in the message body. The content type "application/x-www-form-urlencoded" is used for normal form data, submitted using the POST method. The content length indicates the exact length of the string to be read. This information is the same when the form is submitted to an HTTP server, and is available to to the CGI program.

The URL encoding of the form data is fairly easy to decode, at least for a program (it's tedious for humans to do). Each field is listed in sequence, with an "=" separating the field name from its value. These name=value pairs are separated from one another by "&" characters. Any spaces in the data are converted to "+" and any other special characters are encoded in hexadecimal and displayed as a 3-character string (such as "%2C" for a "," character).

Now that we know how the encoding works, we can extract the information from the form, and process it in a program. But, we'll leave that for next time.

Further Reading

A good, although brief, description of fill-out forms can be found at NCSA.

[http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html]

Ian Graham's Introduction to HTML also has a good description of forms.

[http://www.hprc.utoronto.ca/HTMLdocs/NewHTML/forms.html]

Finally, users of Netscape may want to check the release notes for their version, for possible form-related extensions.

[http://home.netscape.com/eng/mozilla/2.0/relnotes/]

Doom! continued from page 3

- Linux SVGAlib and X executables
- SVGAlib 1.2.0, patched to work better with MouseSystems-type mice
- The shareware DOOM WAD file (version 1.8)
- READMEs for everything, and the SVGAlib patch I applied

SVGAlib 1.2.5, which is current as of this writing, includes the MouseSystems patch; if you have that version or later, you don't need the one included in the linux-doom-1.8 package. An HTML-formatted current index of the files in sunsite's Linux DOOM directory is available in ftp://sunsite.unc.edu/pub/Linux/games/doom/INDEX.html .

ftp.idsoftware.com is id's official FTP site; their files are in directory /idstuff. The latest updates to MS-DOS DOOM appear there first. A large archive of user-contributed WAD files and other goodies are available under the / tonsmore directory. The same material is also available by FSP at fsp.idsoftware.com. If you want to buy the registered copy of DOOM for MS-DOS (which you have to do to legally obtain the registered doom.wad file), call 1-800-IDGAMES. The cost is $40 plus (approximately) $6 shipping, charged to a major credit card. Delivery time is usually around a couple of weeks. DOOM II is sold only in stores or by retail mail order.

Prerequisites and installation

To run DOOM under Linux, you will need:

- Linux 1.0 or later
- SVGAlib 1.1.8 or later OR XFree86 2.0 or above (if you compiled it from source yourself, make sure to have the MITSHM extension enabled, and enable SYSV IPC in your Linux kernel)
- The Linux DOOM executables
- A doom*.wad file for version 1.8 of DOOM (which contains all of the game data)

For best performance, you should have a 486-33 or better with at least 8M of memory and a VLB or PCI-bus video card (although many people are getting good results with high-quality ISA-bus video cards as well). There are quite a few variables that affect DOOM performance; some people find it unplayable on otherwise spiffy-looking systems and some 386-40 users think it's perfectly playable on their systems.

Next month we continue this fascinating look at fun and games on Linux with a bit about what you need for getting sound out of your Linux box during the game! Stay tuned, fans of blood 'n' gore!