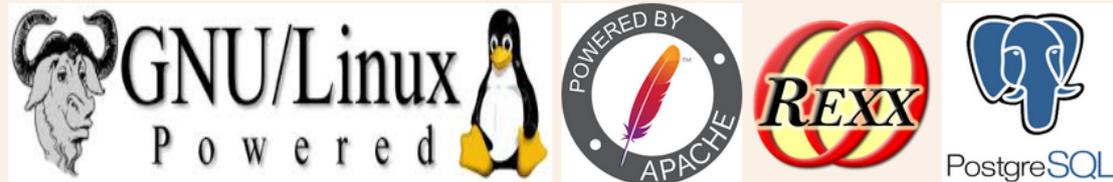


# Writing Web CGI with REXX

Using REXX as a programming language for writing Web CGI applications.



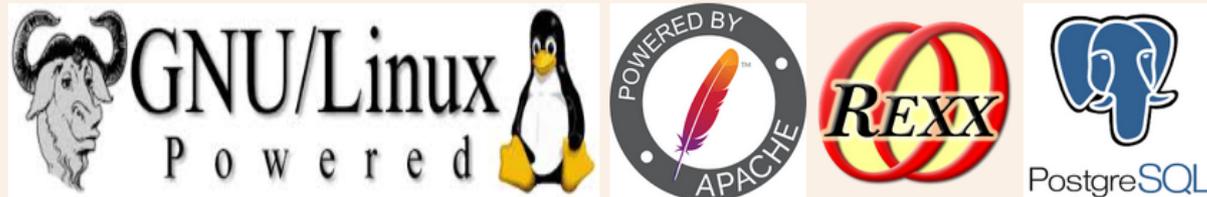
# Writing Web CGI with REXX

This presentation / tutorial is a continuation of the presentation given by the author at the 2023 REXX Symposium Titled “Full Stack Development with REXX.”

[https://www.rexxla.org/presentations/2023/full\\_stack\\_rexx.pdf](https://www.rexxla.org/presentations/2023/full_stack_rexx.pdf)

That presentation was an overview of using REXX as a programming language for end-to-end application development.

This presentation / tutorial focuses on using REXX to write Web CGI for creating dynamic web content and GUI based web applications.

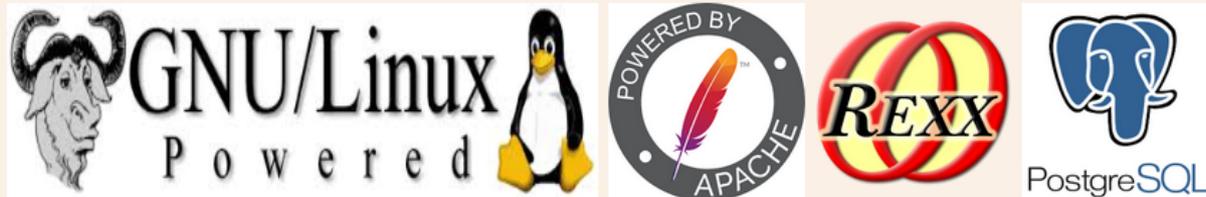


# Writing Web CGI with REXX

The author has seen posts over the years of REXX developers asking for access to a “REXX enabled web server” to serve REXX CGI programs.

In the opinion of today’s author / presenter, the best web server to serve REXX CGI programs is the Linux Apache Web server running in YOUR basement.

The author has had a Linux server running in his basement continuously since 1999.

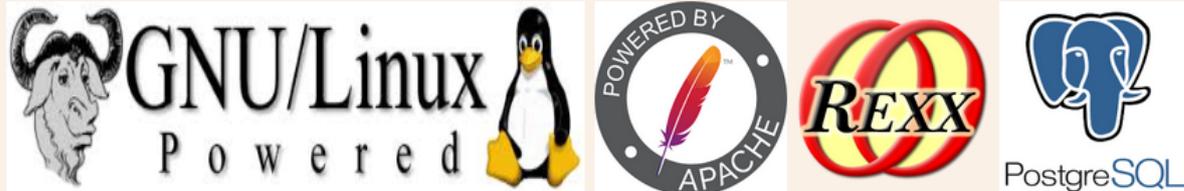


# Writing Web CGI with REXX

## Turn of the century technology

Web CGI has been around since the earliest days of the world wide web and REXX has been used to write CGI programs since the beginning.

This combination has proven to be stable and adaptable.



# Writing Web CGI with REXX

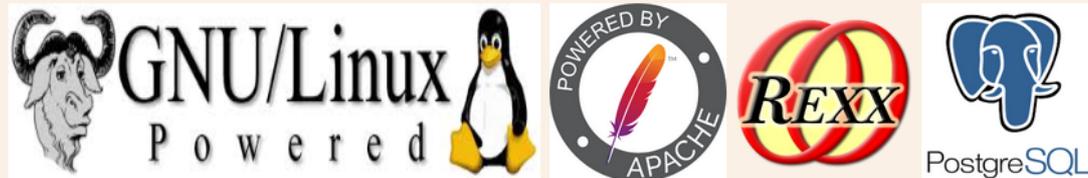
## Getting Started

What we need.

In the opinion of the author, the best environment to run CGI programs in REXX today is a Linux server running the Apache web server.

It is very affordable today to purchase a small cloud server running debian or some other linux distribution.

The hardware requirements are likewise minimal to get started. A surplus desktop computer with internet access will do just fine. The set-up / configuration and administration of a Linux Web server is beyond the scope of this presentation, but we will try to cover some of the Apache configuration basics.



# Writing Web CGI with REXX

## Getting Started

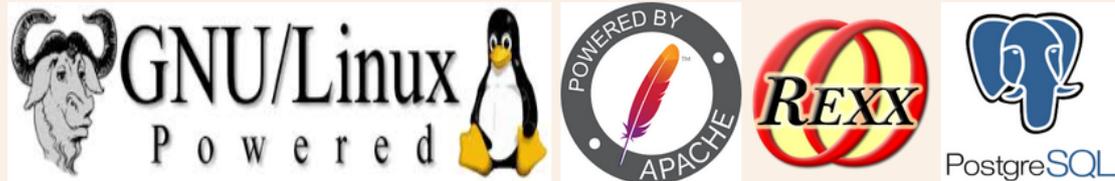
What we need.

Basic knowledge of HTML is a must.

Knowledge of CSS (Cascading Style Sheets) is extremely useful.

Some knowledge of javascript can also be useful for adding client side content.

A working knowledge of the REXX programming language.



# Writing Web CGI with REXX

Start with simple HTML (Right out of Laura LeMay!)

In 1999, the author moved from a mainframe support group to a Unix support group. The Author found the best way to learn Unix, was to build a Linux box in his basement.



The hardware requirements for a simple Linux Apache Web Server are very minimal. Any modern desktop PC can be re-purposed as a Linux Server.

Cloud based servers with full time internet access are also very affordable.

You also can test inside your local network and not be internet facing.

# Writing Web CGI with REXX

Complete  
HTML  
source for  
original  
home page.  
Circa 1999

```
1 <HEAD>
2   <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html" >
3   <TITLE>Welcome to lrschacher.com !</TITLE>
4 </HEAD>
5 <style>
6 body {
7   background-color: faf0e6;
8 }
9 h1 {
10  text-align: center;
11 }
12
13 h2 {
14  text-align: center;
15 }
16 p {
17  text-align: center;
18  font-family: verdana;
19  font-size: 20px;
20 }
21 </style>
22 <BODY>
23 <H1 ALIGN=CENTER> </H1>
24 <H1>The Larry Schacher Personal Enrichment Project</H1>
25 <H1 ALIGN=CENTER><IMG SRC="/graphics/cts1.gif" NAME="Graphic1" ALIGN=BOTTOM WIDTH=120 HEIGHT=83 BORDER=0></H1>
26 <P ALIGN=CENTER>&quot;The formula for success is where preparation meets opportunity&quot;</P>
27 <P ALIGN=CENTER><IMG SRC="/graphics/const1.gif" NAME="Graphic2" ALIGN=ABSMIDDLE WIDTH=50 HEIGHT=74 BORDER=0><STRONG>Under construction. Please pardon our dust !</STRONG></P>
28 <HR SIZE=8>
29 <H2>What is the Enrichment Project?</H2>
30 <P>This site was originally built in order to learn UNIX systems in general and LINUX in particular.</P>
31 <H2>Why was it built?</H2>
32 <P>It was thought that instead of simply learning UNIX in the abstract, it would be nice to actually build a working system.</P>
33 <HR SIZE=8>
34 <BR>
35 <CENTER>
36 <A HREF="http://www.gnu.org/" target="_blank"><IMG SRC="/graphics/gnu1.jpg" NAME="Graphic1" width="240" height="80"> /A>
37 </CENTER>
38 </BODY>
39 </HTML>
```

A working  
knowledge  
of HTML is  
needed for  
CGI.



# Writing Web CGI with REXX

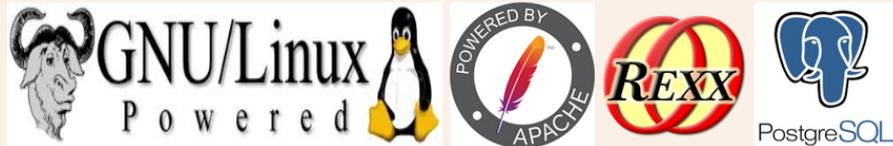
## What is Web CGI? Common Gateway Interface

From Wikipedia, the free encyclopedia.

In computing, Common Gateway Interface (CGI) is an interface specification that enables web servers to execute an external program to process HTTP or HTTPS user requests.

Such programs are often written in a scripting language and are commonly referred to as CGI scripts, but they may include compiled programs.

The author of this presentation has been writing CGI programs in REXX since 1997.



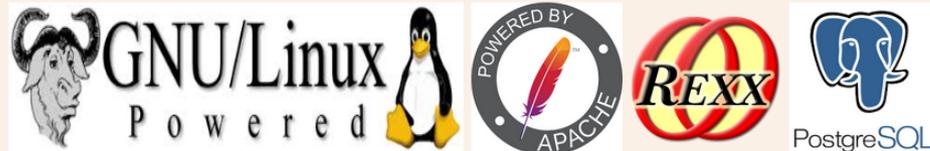
# Writing Web CGI with REXX

Some basic Apache Server configuration.

Below is some basic Apache server configuration to enable serving web pages including execution of CGI from a specified directory on your linux server.

```
AddHandler cgi-script .cgi .pl .py
Alias /home/ "/opt/apps/web1/"
Alias /home "/opt/apps/web1/"
Alias /web1/ "/opt/apps/web1/"
Alias /web1 "/opt/apps/web1/"
<Directory "/opt/apps/web1">
  Options Indexes ExecCGI FollowSymLinks MultiViews
  AllowOverride None
  Order allow,deny
  Allow from all
  Require all granted
</Directory>
```

Note: Detailed Apache Server administration is beyond the scope of this presentation.



# Writing Web CGI with REXX

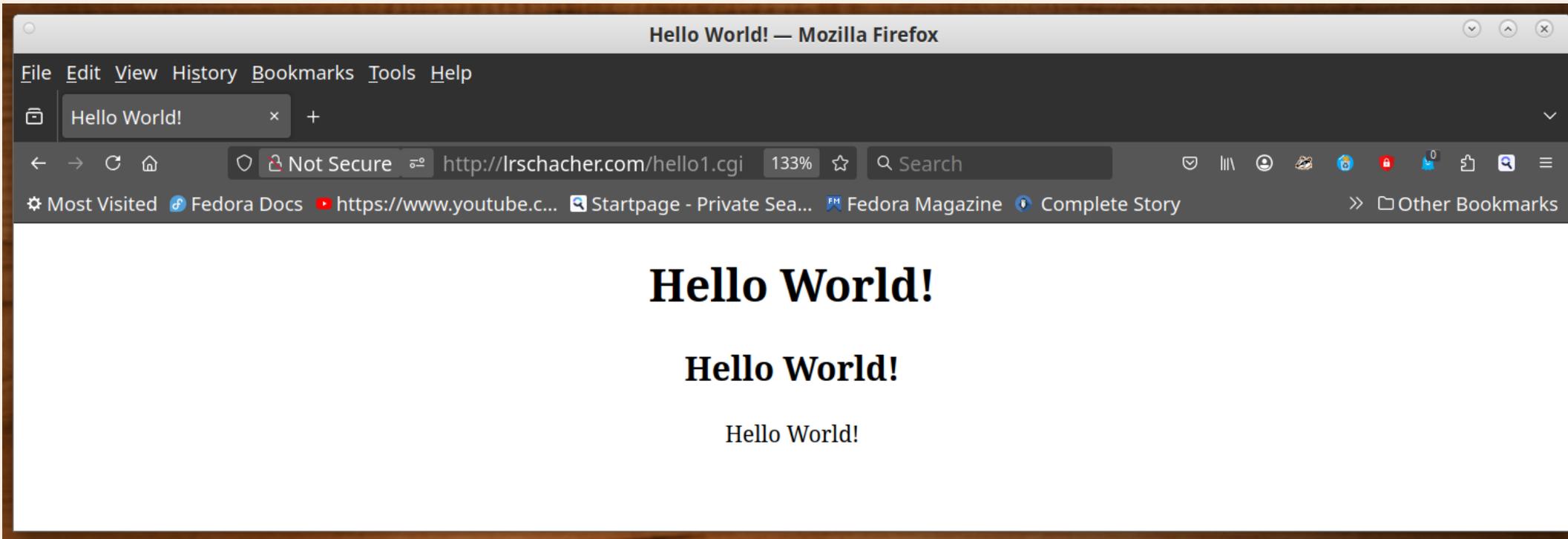
Our first CGI program in REXX.

```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal x Mate Terminal x
/opt/apps/web1/hello1.cgi Size=20 Line=0 Col=1 Alt=1,1
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+
00000 *** Top of File ***
00001 #!/usr/local/bin/rexx
00002 /*-----*/
00003 /* Our First CGI program in REXX. We SAY our HTML! */
00004 /*-----*/
00005 SAY 'Content-type: <!DOCTYPE html>'
00006 SAY ''
00007 SAY '<HTML>'
00008 SAY '<HEAD>'
00009 SAY '<CENTER>'
00010 SAY '<TITLE> Hello World! </TITLE>'
00011 SAY '</HEAD>'
00012 SAY '<BODY>'
00013 SAY '<H1> Hello World! </H1>'
00014 SAY '<H2> Hello World! </H2>'
00015 SAY '<P> Hello World! </P>'
00016 SAY '</BODY>'
00017 SAY '</CENTER>'
00018 SAY '</HTML>'
00019 RETURN 0
00020
00021 *** Bottom of File ***
====>
THE 4.0 Files=1 Width=1000 CO
```

We SAY our HTML!



# Writing Web CGI with REXX



# Writing Web CGI with REXX

The output of a REXX CGI program is HTML. The REXX source code is not visible to the client.

This is referred to as “server side programming”

The actual REXX code is executed on the server.

```
1 <HTML>
2 <HEAD>
3 <CENTER>
4 <TITLE> Hello World! </TITLE>
5 </HEAD>
6 <BODY>
7 <H1> Hello World! </H1>
8 <H2> Hello World! </H2>
9 <P> Hello World! </P>
10 </BODY>
11 </CENTER>
12 </HTML>
13
```

Languages like javascript are referred to as “client side programming” and the code is executed on the client’s web browser and therefore, the source code is available to be seen in the client’s web browser.



# Writing Web CGI with REXX

Hey! What went wrong!

Debugging CGI (in any programming language) can be tricky!

## Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at [lschacher@yahoo.com](mailto:lschacher@yahoo.com) to inform them of the time this error occurred, and the actions you performed just before this error.

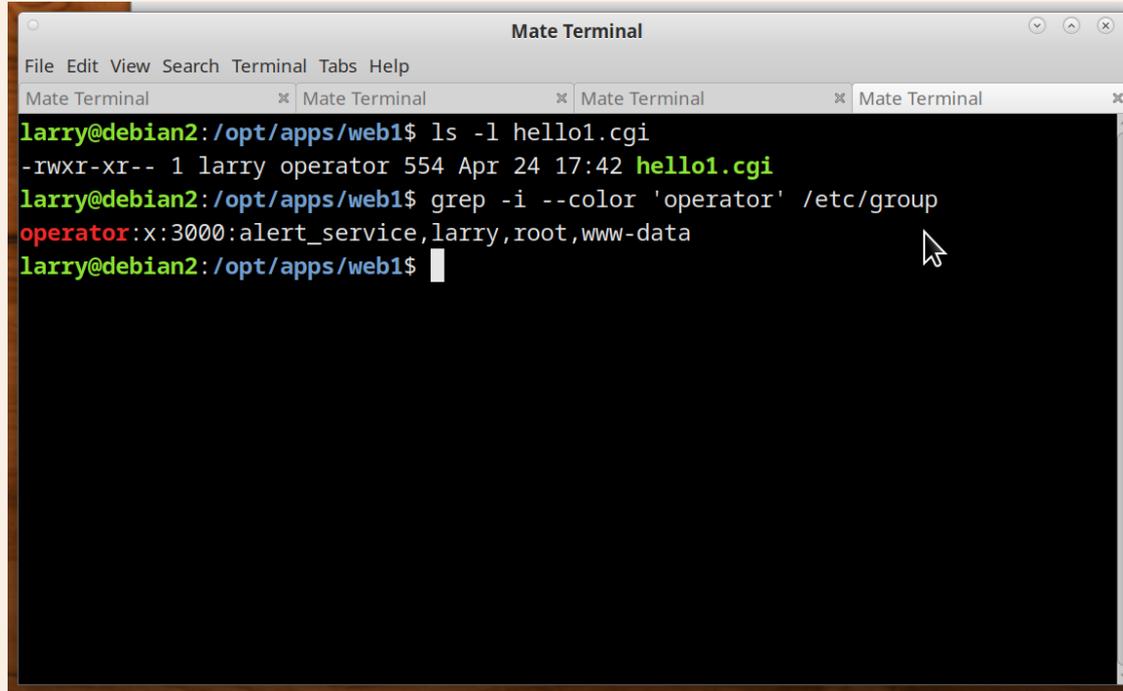
More information about this error may be available in the server error log.



# Writing Web CGI with REXX

Here are some simple debugging suggestions.

First, make sure the cgi program is executable

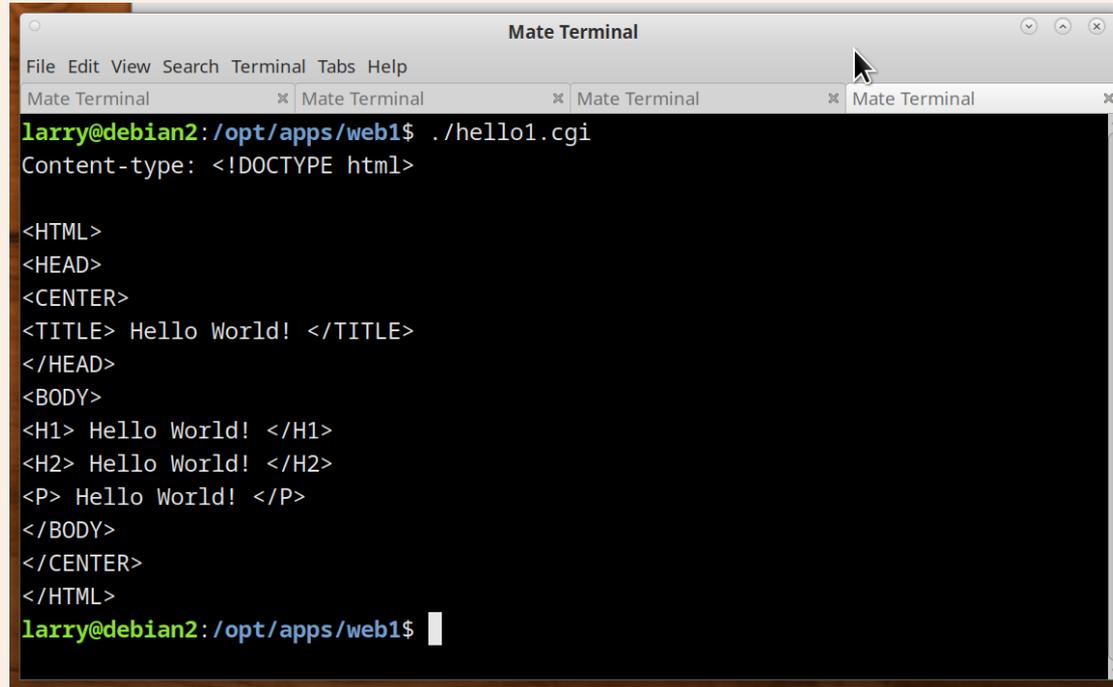


```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal x Mate Terminal x
larry@debian2:/opt/apps/web1$ ls -l hello1.cgi
-rwxr-xr-- 1 larry operator 554 Apr 24 17:42 hello1.cgi
larry@debian2:/opt/apps/web1$ grep -i --color 'operator' /etc/group
operator:x:3000:alert_service,larry,root,www-data
larry@debian2:/opt/apps/web1$
```

Security is an important consideration. Make sure that the Apache Daemon is not running as root. In this case user www-data is in the operator group

# Writing Web CGI with REXX

Here are some simple debugging suggestions.



```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal x Mate Terminal x
larry@debian2: /opt/apps/web1$ ./hello1.cgi
Content-type: <!DOCTYPE html>

<HTML>
<HEAD>
<CENTER>
<TITLE> Hello World! </TITLE>
</HEAD>
<BODY>
<H1> Hello World! </H1>
<H2> Hello World! </H2>
<P> Hello World! </P>
</BODY>
</CENTER>
</HTML>
larry@debian2: /opt/apps/web1$
```

You can test the REXX program from the terminal. Standard REXX errors will usually show up and can be found and fixed.



# Writing Web CGI with REXX

Here are some simple debugging suggestions.  
You can also view the output of the apache error.log

```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal x Mate Terminal
root@debian2:/var/log/apache2# tail error.log
[Thu Apr 24 16:20:12.977255 2025] [cgi:error] [pid 787:tid 873] [client 192.168.1.193:40688] malformed header from script 'hello1.cgi': Bad header: <!DOCTYPE html>
[Thu Apr 24 16:20:13.951762 2025] [cgi:error] [pid 788:tid 842] [client 192.168.1.193:40692] malformed header from script 'hello1.cgi': Bad header: <!DOCTYPE html>
[Thu Apr 24 16:20:17.443484 2025] [cgi:error] [pid 787:tid 876] [client 192.168.1.193:53662] malformed header from script 'hello1.cgi': Bad header: <!DOCTYPE html>
[Thu Apr 24 16:21:06.631388 2025] [cgi:error] [pid 787:tid 872] [client 192.168.1.193:51074] malformed header from script 'hello1.cgi': Bad header: <!DOCTYPE html>
[Thu Apr 24 16:29:14.746121 2025] [cgi:error] [pid 787:tid 880] [client 192.168.1.193:57422] malformed header from script 'hello1.cgi': Bad header: <HTML>
[Thu Apr 24 16:29:21.341939 2025] [cgi:error] [pid 787:tid 875] [client 192.168.1.193:46120] malformed header from script 'hello1.cgi': Bad header: <HTML>
[Thu Apr 24 16:33:23.871233 2025] [cgi:error] [pid 787:tid 867] [client 192.168.1.193:41764] malformed header from script 'hello1.cgi': Bad header: <HTML>
[Thu Apr 24 17:38:59.171350 2025] [cgi:error] [pid 787:tid 867] [client 192.168.1.193:37278] malformed header from script 'hello1.cgi': Bad header: <HTML>
[Thu Apr 24 17:39:00.207495 2025] [cgi:error] [pid 787:tid 861] [client 192.168.1.193:37290] malformed header from script 'hello1.cgi': Bad header: <HTML>
[Thu Apr 24 17:39:02.570801 2025] [cgi:error] [pid 787:tid 872] [client 192.168.1.193:37292] malformed header from script 'hello1.cgi': Bad header: <HTML>
root@debian2:/var/log/apache2#
```



# Writing Web CGI with REXX

## SLAC's REXX WWW CGI Function Library

Last Update: 3 Mar 1997. URL=<http://www.slac.stanford.edu/slac/www/tool/cgi-rexx/>

To call the following functions from your script you will need to include the following in your script:

```
CALL PUTENV 'REXXPATH=/afs/slac/www/slac/www/tool/cgi-rexx'
```

### Index of REXX CGI Functions

Function	Owner	Group	Bytes	Updated	Comment
<a href="#">testfinger</a>	cottrell	sf	1018	Nov 11 18:06	Example of a script to provide a finger function
<a href="#">minimal</a>	cottrell	sf	459	Mar 3 1996	Simple illustration of a Form CGI Script
<a href="#">testinput</a>	Mwww	oh	1306	Mar 1 1996	Example to show processing of input
<a href="#">cleanquery</a>	cottrell	sf	707	Feb 21 18:37	Removes all occurrences of unassigned variables from CGI query string
<a href="#">cgierror</a>	cottrell	sf	524	Nov 11 18:04	Reports an error and returns
<a href="#">cgidie</a>	cottrell	sf	535	Mar 2 1996	Reports an error and Exits
<a href="#">chkpwd</a>	cottrell	sf	1664	Nov 11 18:06	Check a username/password combination
<a href="#">delquery</a>	cottrell	sf	904	Mar 3 15:29	Remove item from CGI query string
<a href="#">deweb</a>	cottrell	sf	1549	Nov 11 18:06	Converts ASCII Hex coded %XX to ASCII characters
<a href="#">formatdate</a>	cottrell	sf	1344	Feb 21 18:37	Parses the date expression given and returns in Oracle format
<a href="#">fullurl</a>	cottrell	sf	531	Feb 21 18:37	Returns the complete CGI query URL
<a href="#">getowner</a>	cottrell	sf	384	Feb 21 18:36	Returns owner of a specified file
<a href="#">getfullhost</a>	cottrell	sf	414	Feb 21 19:26	Returns the fully qualified domain name of the local host
<a href="#">htmlbreak</a>	cottrell	sf	785	Feb 21 18:37	Breaks a long line into lines appropriate for HTML parsing
<a href="#">htmlbot</a>	cottrell	sf	135	Jan 20 1996	Insert boiler plate at end of page
<a href="#">htmltop</a>	cottrell	sf	305	Nov 11 18:19	Insert title and h1 header at top of page
<a href="#">httab</a>	cottrell	sf	2991	Nov 11 18:06	Convert a tab delimited file to an HTML table
<a href="#">methget</a>	cottrell	sf	153	Nov 21 1995	Returns true if the form is using METHOD="GET"
<a href="#">methpost</a>	cottrell	sf	158	Nov 21 1995	Returns true if the form is using METHOD="POST"
<a href="#">myurl</a>	cottrell	sf	239	Nov 11 18:06	Adds the URL of the script to the page
<a href="#">oraenv</a>	crane	bs	656	Feb 7 1996	Sets up the SLAC Oracle/REXX environment
<a href="#">printhead</a>	cottrell	sf	1192	Feb 18 15:02	Inserts the Content-type header
<a href="#">printvariables</a>	cottrell	sf	629	Mar 3 1996	Adds a listing of the Form name=value& variables to the page
<a href="#">readform</a>	cottrell	sf	531	Jan 26 1996	Reads a Form's "GET" or "POST" input and returns it decoded
<a href="#">readpost</a>	cottrell	sf	1697	Nov 11 18:06	Reads the standard input from a form with METHOD="POST"
<a href="#">slacfno</a>	cottrell	sf	1711	Nov 11 18:06	Identifies the allowed visibility of a file
<a href="#">striptml</a>	cottrell	sf	618	Feb 21 18:36	Removes HTML markup from an input string
<a href="#">suspect</a>	cottrell	sf	555	Nov 11 18:06	Provides an error message if the input string contains a suspect character
<a href="#">webify</a>	cottrell	sf	1038	Nov 11 18:06	Encodes special characters in hex ASCII %XX form
<a href="#">wraplines</a>	cottrell	sf	716	Feb 21 18:36	Breaks long lines into lines appropriate for terminal output

[Les Cottrell](#) | [Feedback](#)

Much of Les Cottrell's CGI Function library also runs just fine with ooRexx 5 on Apache 2.4 with only minor modifications

Sadly, shortly after my 2023 presentation, SLAC has removed the REXX WWW CGI Function Library.

The library and many of the articles have been preserved on Howard Fosdick's rexxinfo.org site.



# Writing Web CGI with REXX

```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal
/opt/apps/web1/index.cgi Size=18 Line=0 Col=1 Alt=0,0
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.]...+...8...+...9...+...
==== *** Top of File ***
==== #!/usr/local/bin/rexx
==== /*-----*/
==== /* Front page for lrschacher.com */
==== /*-----*/
==== f = VALUE('PATH', '/usr/bin:/usr/local/bin:/opt/apps/rexx:/opt/apps/cgi-rexx', 'ENVIRONMENT')
==== SAY printhead()
==== SAY htmltop('The Larry Schacher Personal Enrichment Project')
==== rc = style('style2')
==== SAY '<CENTER>'
==== SAY '<P> Press Button to switch to dark Mode </P>'
==== SAY '<P> <input type="button" onclick="window.location.href = ''dark.cgi'';" value="dark"/></P>'
==== fpage = "fpage.txt"
==== DO WHILE LINES(fpage) > 0
====     SAY LINEIN(fpage)
==== END /* do */
==== CALL gbot
==== RETURN 0
====
====>
THE 4.0 Files=1 Width=1000 C0
```

This is the REXX source code for the current lrschacher.com home page. We use functions from the SLAC CGI-REXX package and some additional functions of our own.



# Writing Web CGI with REXX

Current web page using CGI

The Larry Schacher Personal Enrichment Project

Press Button to switch to dark Mode

dark



"The formula for success is where preparation meets opportunity"

 **Under construction. Please pardon our dust !**

---

**What is the Enrichment Project?**

This site was originally built in order to learn UNIX systems in general and LINUX in particular.

**Why was it built?**

It was thought that instead of simply learning UNIX in the abstract, it would be nice to actually build a working system.

---

**Additions and Applications**

[Loan and Mortgage Calculator](#)

[Stock Reports](#) Requires Userid and Password

The link below is of interest to family members!!

[Family Tree \(a perpetual work in progress\)](#) Requires Userid and Password

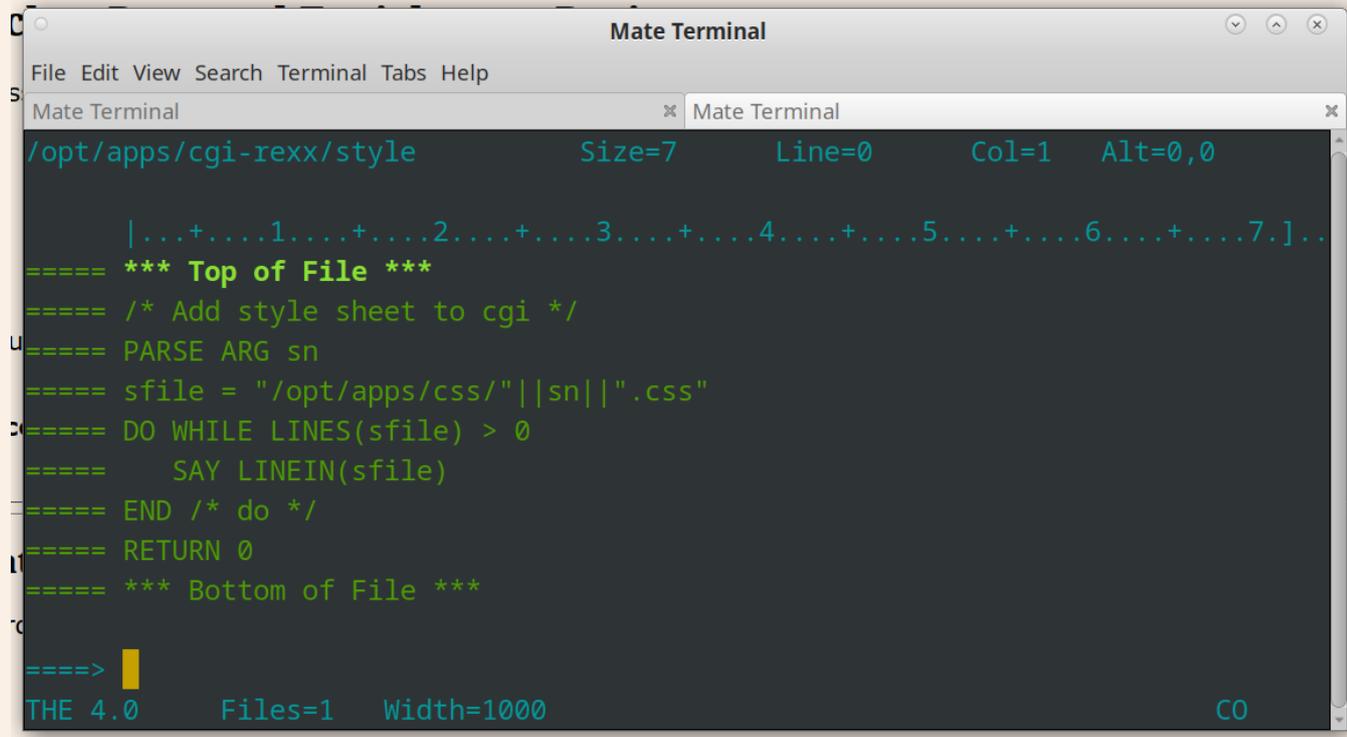
---



# Writing Web CGI with REXX

SAY it with style()

A simple REXX function (created by the author) allows easy incorporation of CSS (Cascading Style Sheets) in your dynamic web content .



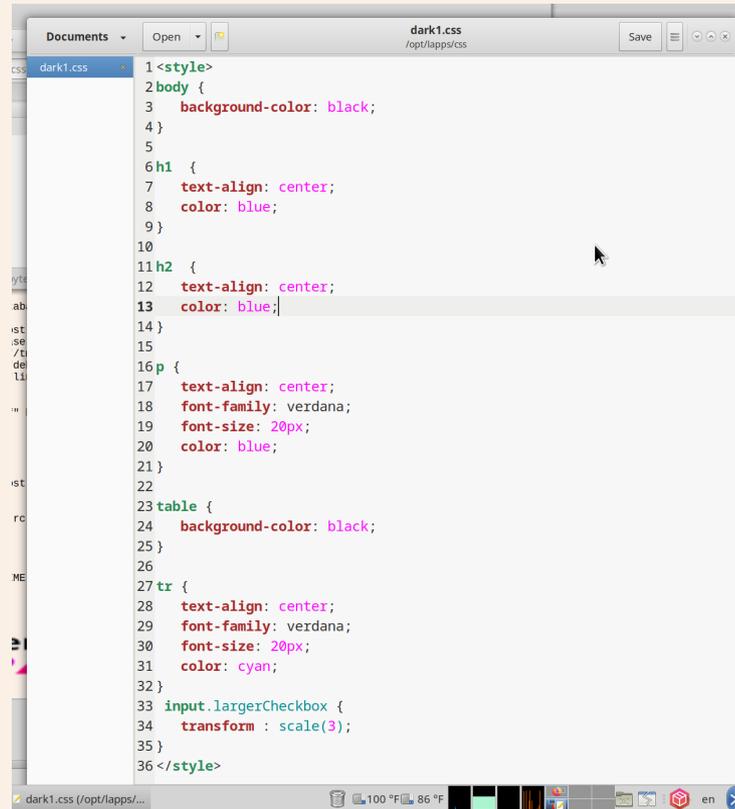
```
File Edit View Search Terminal Tabs Help
Mate Terminal
/opt/apps/cgi-rexx/style      Size=7      Line=0      Col=1      Alt=0,0

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.]..
==== *** Top of File ***
==== /* Add style sheet to cgi */
==== PARSE ARG sn
==== sfile = "/opt/apps/css/"||sn||".css"
==== DO WHILE LINES(sfile) > 0
====     SAY LINEIN(sfile)
==== END /* do */
==== RETURN 0
==== *** Bottom of File ***

====>
THE 4.0      Files=1      Width=1000      CO
```

# Writing Web CGI with REXX

Any Cascading Style Sheet can be included in dynamic Web Content

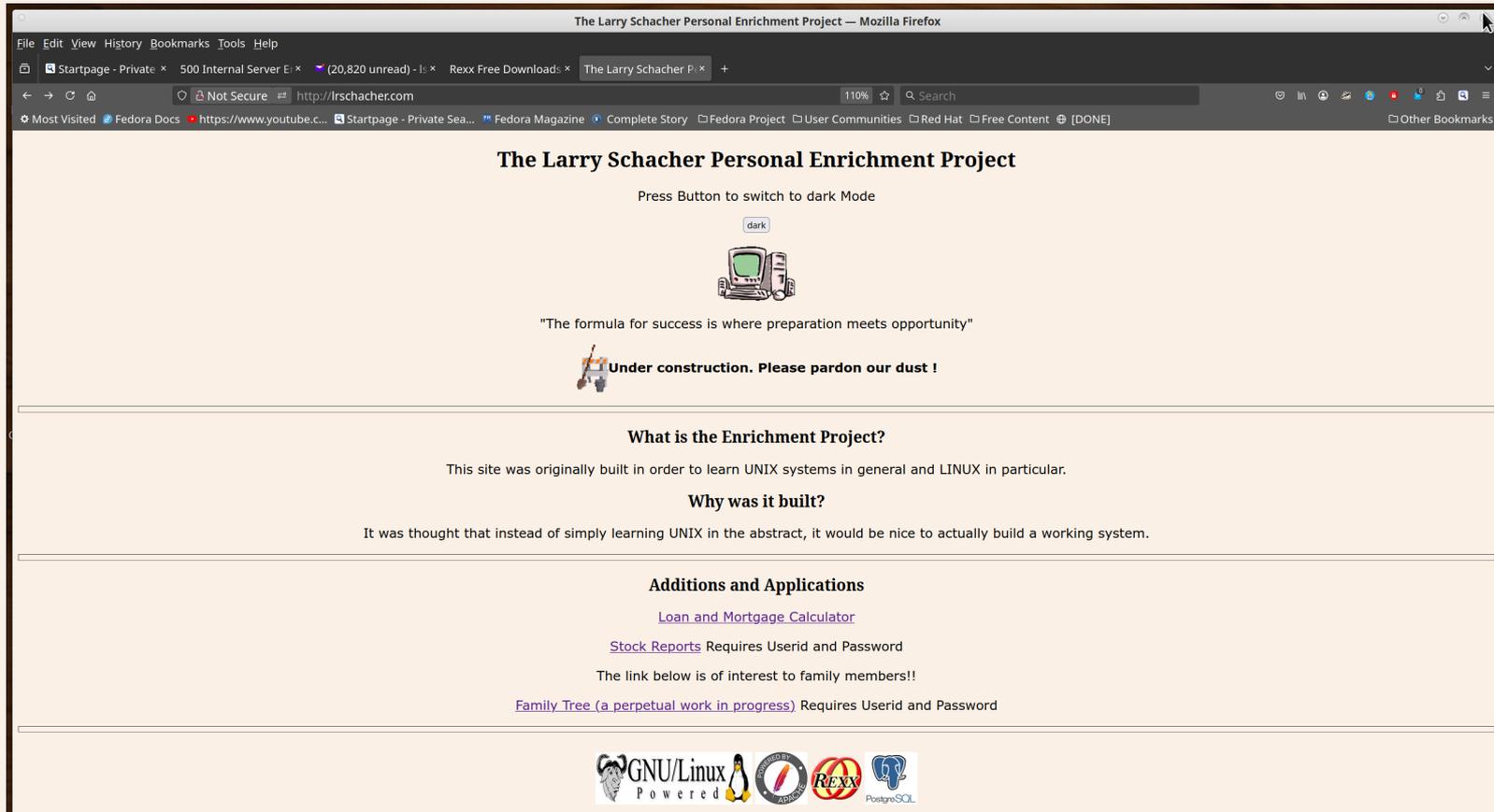


```
1 <style>
2 body {
3   background-color: black;
4 }
5
6 h1 {
7   text-align: center;
8   color: blue;
9 }
10
11 h2 {
12  text-align: center;
13  color: blue;
14 }
15
16 p {
17  text-align: center;
18  font-family: verdana;
19  font-size: 20px;
20  color: blue;
21 }
22
23 table {
24  background-color: black;
25 }
26
27 tr {
28  text-align: center;
29  font-family: verdana;
30  font-size: 20px;
31  color: cyan;
32 }
33 input.largerCheckbox {
34  transform : scale(3);
35 }
36 </style>
```



# Writing Web CGI with REXX

Let's press the dark button!

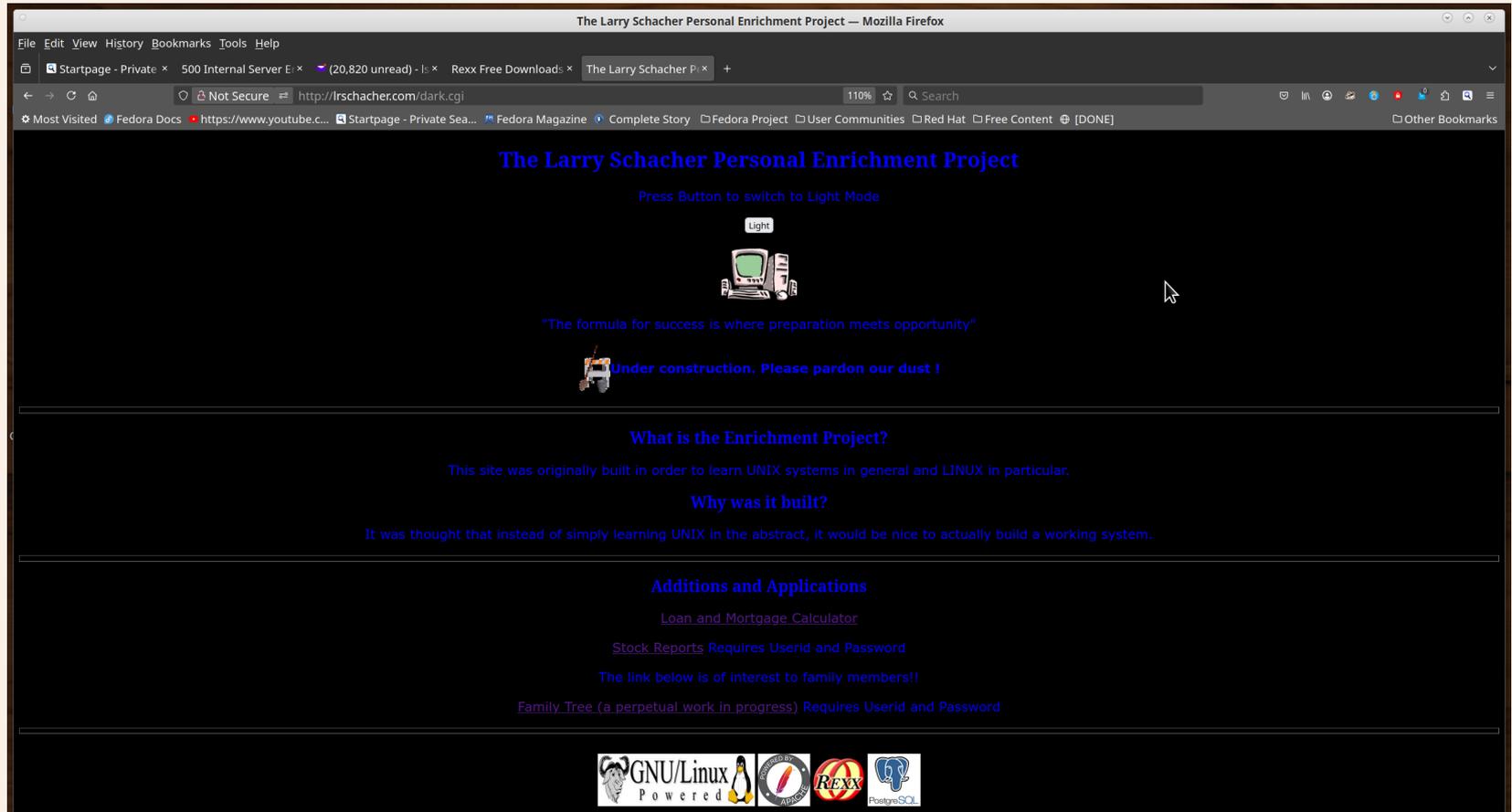


The screenshot shows a Mozilla Firefox browser window with the following content:

- Title Bar:** The Larry Schacher Personal Enrichment Project — Mozilla Firefox
- Address Bar:** Not Secure http://lrschacher.com
- Page Header:** The Larry Schacher Personal Enrichment Project
- Text:** Press Button to switch to dark Mode
- Button:** A button labeled "dark" with a dark theme icon.
- Image:** An illustration of a vintage computer terminal.
- Text:** "The formula for success is where preparation meets opportunity"
- Text:** **Under construction. Please pardon our dust !**
- Section Header:** **What is the Enrichment Project?**
- Text:** This site was originally built in order to learn UNIX systems in general and LINUX in particular.
- Section Header:** **Why was it built?**
- Text:** It was thought that instead of simply learning UNIX in the abstract, it would be nice to actually build a working system.
- Section Header:** **Additions and Applications**
- Text:** [Loan and Mortgage Calculator](#)
- Text:** [Stock Reports](#) Requires Userid and Password
- Text:** The link below is of interest to family members!!
- Text:** [Family Tree \(a perpetual work in progress\)](#) Requires Userid and Password
- Page Footer:** Logos for GNU/Linux Powered, REXX, and PostgreSQL.

# Writing Web CGI with REXX

Easy access to CSS gives your dynamic web content a new look



The screenshot shows a Mozilla Firefox browser window with the title "The Larry Schacher Personal Enrichment Project — Mozilla Firefox". The address bar shows the URL "http://lrschacher.com/dark.cgi". The page content includes:

- Navigation menu: File, Edit, View, History, Bookmarks, Tools, Help
- Browser tabs: Startpage - Private, 500 Internal Server E, (20,820 unread) - I, REXX Free Download, The Larry Schacher P
- Address bar: Not Secure, http://lrschacher.com/dark.cgi, 110%, Search
- Bookmarks bar: Most Visited, Fedora Docs, https://www.youtube.c..., Startpage - Private Sea..., Fedora Magazine, Complete Story, Fedora Project, User Communities, Red Hat, Free Content, [DONE], Other Bookmarks
- Main content:
  - Section: **The Larry Schacher Personal Enrichment Project**
  - Text: Press Button to switch to Light Mode
  - Button: Light
  - Image: A computer monitor and keyboard icon.
  - Quote: "The formula for success is where preparation meets opportunity"
  - Image: A construction worker icon.
  - Text: **Under construction. Please pardon our dust !**
- Section: **What is the Enrichment Project?**
- Text: This site was originally built in order to learn UNIX systems in general and LINUX in particular.
- Section: **Why was it built?**
- Text: It was thought that instead of simply learning UNIX in the abstract, it would be nice to actually build a working system.
- Section: **Additions and Applications**
- Text: [Loan and Mortgage Calculator](#)
- Text: [Stock Reports Requires Userid and Password](#)
- Text: The link below is of interest to family members!!
- Text: [Family Tree \(a perpetual work in progress\)](#) Requires Userid and Password

At the bottom of the page, there are logos for GNU/Linux Powered, REXX, and PostgreSQL.

# Writing Web CGI with REXX

## Processing User Input

### Basic HTML Form Example

Pressing the Submit button on this simple HTML form will POST your details and be processed by another part of the application and shown on a results page.

```
<form name='demoform' method='POST' action='fr1.cgi'>
```

Username: <input type="text"/>
Password: <input type="password"/>
TextArea Comment: Comments ... <input type="text"/>
Filename: <input type="button" value="Browse..."/> No file selected.
Checkbox Items: <input type="checkbox"/> Checkbox 1 <input type="checkbox"/> Checkbox 2 <input checked="" type="checkbox"/> Checkbox 3
Radio Items: <input type="radio"/> radio 1 <input checked="" type="radio"/> radio 2 <input type="radio"/> radio 3
Multiple Select Values <input type="text" value="Selection Item 1"/> <input type="text" value="Selection Item 2"/> <input type="text" value="Selection Item 3"/> <input type="text" value="Selection Item 4"/>
Dropdown: <input type="text" value="Drop Down Item 3"/> ▾
<input type="button" value="reset"/> <input type="button" value="submit"/>

The HTML <FORM> tag and its associated elements allow us to “draw” our GUI elements on a Web Page.

# Writing Web CGI with REXX

REXX source for creating the HTML form

```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal
/opt/apps/web1/form1.cgi Size=15 Line=0 Col=1 Alt=0,0

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.]...+...8...+...9...+...10...+...11...+...12...+...
==== ** Top of File **
==== #!/usr/local/bin/rexx
==== /*-----*/
==== /* Testing HTML 5 */
==== /*-----*/
==== f = VALUE('PATH', '/usr/bin:/usr/local/bin:/opt/apps/rexx:/opt/apps/cgi-rexx', 'ENVIRONMENT')
==== SAY printhead('!DOCTYPE html')
==== SAY htmltop('Processing User Input')
==== rc = style('style2')
==== fpage = "form1.txt"
==== DO WHILE LINES(fpage) > 0
====     SAY LINEIN(fpage)
==== END /* do */
==== CALL gbot
==== RETURN 0
====
==== ** Bottom of File **

====>
THE 4.0 Files=1 Width=1000 CO
```



# Writing Web CGI with REXX

HTML tables are often used to align elements on a web page

```
1 <center>
2 <h2>Basic HTML Form Example</h2>
3 <h3>Pressing the Submit button on this simple HTML form will POST your details <br>
4 and be processed by another part of the application and shown on a results page.
5 </h3>
6 <form name='demoform' method='POST' action='f01.cgi'>
7 <table border="1" cellpadding="5">
8 <tr>
9 <td> Username:<br> <input type="text" name="username" size="15" /> </td>
10 </tr>
11 <tr>
12 <td> Password:<br> <input type="password" name="password" size="15" /> </td>
13 </tr>
14 <tr>
15 <td> TextArea Comment:<br> <textarea cols="40" name="comments" rows="6"> Comments...</textarea> </td>
16 </tr>
17 <tr>
18 <td> Filename:<br> <input type="file" name="filename" size="35" /> </td>
19 </tr>
20 <tr>
21 <td>
22 <div>
23 <input type="checkbox" name="checkbox01" value="cb1" />Checkbox 1
24 <input type="checkbox" name="checkbox01" value="cb2" />Checkbox 2
25 <input type="checkbox" name="checkbox01" value="cb3" checked="checked" />Checkbox 3
26 </div>
27 </tr>
28 <tr>
29 <td>
30 <div>
31 <input type="radio" name="radioval" value="rd1" />radio 1
32 <input type="radio" name="radioval" value="rd1" checked="checked" />radio 2
33 <input type="radio" name="radioval" value="rd3" />radio 3
34 </div>
35 </tr>
36 <tr>
37 <td>
38 <div>
39 <select multiple="multiple" name="multiselect1" size="4">
40 <option value="ms1"> Selection Item 1 </option>
41 <option value="ms2"> Selection Item 2 </option>
42 <option value="ms3"> Selection Item 3 </option>
43 <option value="ms4" selected="selected"> Selection Item 4 </option>
44 </select>
45 </div>
46 </tr>
47 <tr>
48 <td>
49 <div>
50 <select name="dropdown">
51 <option value="dd1"> Drop Down Item 1 </option>
52 <option value="dd2"> Drop Down Item 2 </option>
53 <option value="dd3" selected="selected"> Drop Down Item 3 </option>
54 <option value="dd4"> Drop Down Item 4 </option>
55 <option value="dd5"> Drop Down Item 5 </option>
56 <option value="dd6"> Drop Down Item 6 </option>
57 </select>
58 </div>
59 </tr>
60 <tr>
61 <td>
62 <input type="reset" name="submitbutton" value="reset" />
63 <input type="submit" name="submitbutton" value="submit" />
64 </td>
65 </tr>
66 </table>
67 </form>
68 </center>
```

Body of the HTML form



# Writing Web CGI with REXX

## Processing User Input from HTML Form

### Form Contents

USERNAME had a value of larry

PASSWORD had a value of test1

COMMENTS had a value of Comment1

FILENAME had a value of blocked.cgi

CHECKBOX1 had a value of cb3

RADIOVAL had a value of rd2

MULTSELECT1 had a value of ms4

DROPDOWN had a value of dd3

SUBMITBUTTON had a value of submit



# Writing Web CGI with REXX

Part one of REXX program to capture and decode user input from an HTML form

```
opt/apps/web1/fr1.cgi
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...
==== *** Top of File ***
==== #!/usr/local/bin/rexx
==== /*-----*/
==== /* Processing User Input from HTML forms */
==== /*-----*/
==== f = VALUE('PATH', '/usr/bin:/usr/local/bin:/opt/apps/rexx:/opt/apps/cgi-rexx', 'ENVIRONMENT')
==== SAY printhead('<DOCTYPE html>')
==== SAY htmltop('Processing User Input from HTML Form')
==== rc = style('style2')
==== env = "ENVIRONMENT"
==== i=0
==== SAY "<H1>Form Contents</H1>"
==== SAY '<CENTER>'
==== method = value("REQUEST_METHOD", env)
==== len = value("CONTENT_LENGTH", env)
==== IF (method == "GET") THEN
==== DO
====   query_string = value("QUERY_STRING", env)
==== END /* do */
==== IF (method == "POST") & (len >> "") THEN
==== DO
====   /* use POST method to pass parameters */
====   post_string = charin(, len)
====   query_string = post_string
==== END /* do */
====
==== NF = ParseQueryString( query_string) /* NF holds the # of fields */
==== Do j=1 to NF
====   SAY "<P>" Parms.Tag.j "had a value of" Parms.XVal.j "<P>"
==== End
==== SAY '</CENTER>'
==== CALL gbot
==== RETURN 0
```

This REXX program was adapted from an old OS/2 REXX program called CGIPARSE.COMD



# Writing Web CGI with REXX

Part two of REXX program to capture and decode user input from an HTML form

```
/opt/apps/web1/fr1.cgi
=====
[...+...1...+...2...+...3...+...4...+...5...+...6...+...7...]
===== /* Do not modify below this line -- Important parsing code...
===== From Frankie Fan's OS2HTTPD archive
===== */
===== ParseQueryString: procedure expose Parms, NFields
=====   Parse arg P
=====   i = 1
=====   do while ((P \= '') & (i < 1000))
=====     Parse var P Parms.Text.i '&' rest
=====     Parse var Parms.Text.i Parms.Tag.i '=' Parms.KeyVal.i
=====     Parms.Tag.i = translate( Parms.Tag.i)
=====     Parms.XVal.i=DecodeKeyVal( Parms.KeyVal.i)
=====     P = rest
=====     i = i + 1
=====   end
=====   NFields = i - 1
=====   return NFields
=====
===== DecodeKeyVal: procedure
=====   parse arg Code
=====   Text= ''
=====   Code=translate(Code, ' ', '+') /* Convert + signs to spaces */
=====   rest='% '
=====   do while (rest\='')
=====     Parse var Code T '% ' rest
=====     Text=Text || T
=====     if (rest\='') then
=====       do
=====         ch = left( rest,2)
=====         c=X2C(ch) /* Hex to character conversion */
=====         Text=Text || c
=====         Code=substr( rest, 3)
=====       end
=====     end
=====   end
=====   return Text
===== *** Bottom of File ***
```

This REXX program was adapted from an old OS/2 REXX program called CGIPARSE.COMD



# Writing Web CGI with REXX

The Advanced!

A complete GUI application in the WEB

## Loan and Mortgage Calculator

Dark Mode

Requested Loan Amount: \$

Interest Rate:

Years to Repay:

Results:

Display Amortization:

This program is an example of a recursive CGI program



# Writing Web CGI with REXX

```

Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal
/opt/apps/web1/loan1.cgi
Size=179 Line=0 Col=1 Alt=0,0
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
==== ** Top of File **
==== #!/usr/local/bin/rexx
==== /*-----*/
==== /* loan program */
==== /*-----*/
==== f = VALUE('PATH', '/usr/bin:/usr/local/bin:/opt/apps/rexx:/opt/apps/cgi-rexx', 'ENVIRONMENT')
==== CALL cgiparse
==== payment = ""
==== totpaid = ""
==== totintr = ""
==== crlf = "&#13;&#10;"
==== SAY printhead()
==== SAY htmltop('Loan and Mortgage Calculator')
==== IF darkmode = 1 THEN rc1 = STYLE('dark1')
==== ELSE rc1 = STYLE('style4')
==== SAY '<FORM NAME=topform METHOD=POST ACTION="loan1.cgi">'
==== SAY '<CENTER>'
==== DO WHILE POS(",", amount) > 0
====     PARSE VAR amount first ", " last
====     amount = first||last
==== END/* do while */
==== IF darkmode = 1 THEN SAY '<P> <input type="checkbox" name="darkmode" value="1" checked> Dark Mode <INPUT TYPE="SUBMIT" NAME="Set" VALUE="Set">'
==== ELSE SAY '<P> <input type="checkbox" name="darkmode" value="1"> Dark Mode <INPUT TYPE="SUBMIT" NAME="Set" VALUE="Set"> </P>'
==== IF DATATYPE(amount) \= "NUM" & amount \= "AMOUNT" THEN error_string = amount
==== ELSE error_string = ""
==== IF DATATYPE(amount) = "NUM" THEN amt = amount
==== ELSE amt = ""
====>
THE 4.0 Files=1 Width=1000 CO
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal x Mate Terminal x Mate Terminal
/opt/apps/web1/loan1.cgi Size=179 Line=27 Col=1 Alt=0,0
===== ELSE amt = ""
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
===== SAY '<P> Requested Loan Amount: $ <INPUT type="text" name="amount" value = "||amt||"> </P>'
===== IF DATATYPE(interest) = "NUM" THEN int1 = interest
===== ELSE int1 = '5'
===== SAY '<P> Interest Rate:'
===== SAY '<SELECT name="interest">'
===== DO int2 = 1 to 9
===== IF int2 = int1 THEN SAY '<OPTION SELECTED VALUE="||int1||">||int1||'</OPTION>'
===== ELSE SAY '<OPTION VALUE="||int2||">||int2||'</OPTION>'
===== END /* do */
===== SAY '</SELECT>'
===== SAY '</P>'
===== IF DATATYPE(years) = "NUM" THEN yrs1 = years
===== ELSE yrs1 = 5
===== SAY '<P> Years to Repay:'
===== SAY '<SELECT name="years">'
===== DO yrs2 = 1 to 30
===== IF yrs2 = yrs1 THEN SAY '<OPTION SELECTED VALUE="||yrs1||">||yrs1||'</OPTION>'
===== ELSE SAY '<OPTION VALUE="||yrs2||">||yrs2||'</OPTION>'
===== END /* do */
===== SAY '</SELECT>'
===== IF DATATYPE(amount) = "NUM" THEN CALL loan1
===== SAY '<P> Results:</P>'
===== SELECT
===== WHEN output = "Y" THEN SAY "<P><TEXTAREA NAME='result' ROWS=3 COLS=36 WRAP=VIRTUAL READONLY>||LEFT("Monthly Payment:",17)||fpayment ||crlf|
===== WHEN error_string >> "" THEN SAY "<P><TEXTAREA NAME='result' ROWS=3 COLS=36 WRAP=VIRTUAL READONLY>||error_string||crlf||"is not numeric"||c
===== OTHERWISE SAY "<P><TEXTAREA NAME='result' ROWS=3 COLS=36 WRAP=VIRTUAL READONLY>Enter a loan amount above</TEXTAREA></P>"
===== END /* select */
=====
THE 4.0 Files=1 Width=1000 CO
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```

/opt/apps/web1/loan1.cgi
===== END /* select */
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
===== SAY '</P>'
===== IF DATATYPE(amort) = "NUM" THEN SAY '<P> <input type="checkbox" name="amort" value="1" checked> Display Amortization: </P>'
===== ELSE SAY '<P> <input type="checkbox" name="amort" value="1"> Display Amortization: </P>'
===== SAY '<P> <INPUT TYPE="SUBMIT" NAME="Calculate" VALUE="Calculate">'
===== SAY '<input type="button" onclick="window.location.href = ''loan1.cgi'';" value="Reset"/>'
===== SAY '<P><input type="button" onclick="window.location.href = ''index.cgi'';" value="Home Light Mode"/>'
===== SAY '<input type="button" onclick="window.location.href = ''dark.cgi'';" value="Home Dark Mode"/></P>'
===== IF output = "Y" & amort = 1 THEN /* Amortization Table */
===== DO
===== SAY '<TABLE>'
===== counter = 0
===== CALL ahead
===== DO i = 1 to months
===== counter = counter + 1
===== SAY '<TR >'
===== SAY '<TD>' i '</TD>' '<TD>' fprev_bal.i '</TD>' '<TD>' fmonthly.i '</TD>' '<TD>' fprincipal.i '</TD>' '<TD>' fbal.i '</TD>'
===== IF counter = 10 THEN
===== DO
===== CALL ahead
===== counter = 0
===== END /* do */
===== END /* do i */
===== SAY '</TABLE>'
===== SAY '<P><input type="button" onclick="window.location.href = ''loan1.cgi'';" value="Reset"/> </P>'
===== SAY '<P><input type="button" onclick="window.location.href = ''index.cgi'';" value="Home Light Mode"/>'
===== SAY '<input type="button" onclick="window.location.href = ''dark.cgi'';" value="Home Dark Mode"/></P>'
===== END /* do */
=====
THE 4.0 Files=1 Width=1000
Click to start dragging "Mate Terminal"
CO
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```
Mate Terminal
/opt/apps/web1/loan1.cgi
==== END /* do */
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7.]...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
==== SAY '</FORM>'
==== SAY '</CENTER>'
==== CALL gbot
==== RETURN 0
====
==== loan1:
==== output = "Y"
==== months = years * 12
==== interest = (interest / 100) / 12
==== payment = (interest / (( 1 + interest) ** months - 1) + interest) * amount
==== totpaid = payment * months
==== totintr = totpaid - amount
==== fpayment = money(payment)
==== ftotpaid = money(totpaid)
==== ftotintr = money(totintr)
==== IF amort = 1 THEN
==== DO i = 1 to months
====   mthly = interest * amount
====   principal = payment - mthly
====   prev_bal = amount
====   amount = amount - principal
====   fbal.i = MONEY(amount)
====   fmthly.i = MONEY(mthly)
====   fprincipal.i = MONEY(principal)
====   fprev_bal.i = MONEY(prev_bal)
====   END /* do i */
==== RETURN
====>
THE 4.0 Files=1 Width=1000
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```
File Edit View Search Terminal Tabs Help
Mate Terminal
/opt/apps/web1/loan1.cgi
==== RETURN
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
====
==== ahead:
==== SAY '<TR>'
==== SAY '<TH> MONTH </TH>'
==== SAY '<TH> Previous Balance </TH>'
==== SAY '<TH> Interest </TH>'
==== SAY '<TH> Principal </TH>'
==== SAY '<TH> New Balance </TH>'
==== SAY '</TR>'
==== RETURN
==== cglparse:
====
==== env = "ENVIRONMENT"
====
==== /* Read in the product list */
==== i=0
====
==== method = value("REQUEST_METHOD",,env)
==== len = value("CONTENT_LENGTH",,env)
==== IF (method == "GET") THEN
==== DO
====   query_string = value("QUERY_STRING",,env)
==== END /* do */
==== IF (method == "POST") & (len >< "") THEN
==== DO
====   /* use POST method to pass parameters */
====   post_string = charin(,len)
====>
THE 4.0 Files=1 Width=1000
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```
File Edit View Search Terminal Tabs Help
Mate Terminal
/opt/apps/web1/loan1.cgi
==== post_string = charin(,,len)
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
==== query_string = post_string
==== END /* do */
====
==== NF = ParseQueryString( query_string) /* NF holds the # of fields */
==== Do j=1 to NF
====   variable = Parms.Tag.j||" = "||"||"||Parms.XVal.j||"||"
====   INTERPRET variable
==== End
==== RETURN
====
==== /* Do not modify below this line -- Important parsing code...
====   From Frankie Fan's OS2HTTPD archive
==== */
==== ParseQueryString: procedure expose Parms. NFields
====   Parse arg P
====   i = 1
====   do while ((P \= '') & (i < 1000))
====     Parse var P Parms.Text.i '&' rest
====     Parse var Parms.Text.i Parms.Tag.i '=' Parms.KeyVal.i
====     /*Parms.Tag.i = translate( Parms.Tag.i)*/
====     Parms.XVal.i=DecodeKeyVal( Parms.KeyVal.i)
====     P = rest
====     i = i + 1
====   end
====   NFields = i - 1
====   return NFields
====
====>
THE 4.0 Files=1 Width=1000 C0
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

```
Mate Terminal
File Edit View Search Terminal Tabs Help
/opt/apps/web1/loan1.cgi
====
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...14...
==== DecodekeyVal: procedure
==== parse arg Code
==== Text=''
==== Code=translate(Code, '+', ' ') /* Convert + signs to spaces */
==== rest='% '
==== do while (rest\='')
====   Parse var Code T '% ' rest
====   Text=Text || T
====   if (rest\='') then
====     do
====       ch = left( rest,2)
====       c=X2C(ch) /* Hex to character conversion */
====       Text=Text || c
====       Code=substr( rest, 3)
====     end
====   end
====   return Text
==== RETURN 0
==== *** Bottom of File ***
====>
THE 4.0 Files=1 Width=1000
Click to start dragging "Mate Terminal"
CO
```

The remainder of the presentation will be reviewing the application source code and running the application.



# Writing Web CGI with REXX

## About the Presenter.

Larry has over 40 years of Enterprise IT experience. He started his career as a S/370 Mainframe operator. He first learned Rexx in 1993 on VM/CMS, and has since programmed on just about every Rexx platform. In his spare time, Larry enjoys retro computing and likes to run Legacy IBM operating systems using the Hercules Emulator.  
larryschacher@gmail.com

