

# UniVerse Intranet Access

Ross Morrissey

<http://www.rossmorrisey.com/>

We are constantly being pushed to provide improved access to UniVerse from web browsers. My previous article outlined an easy way to make your existing hard-copy UniVerse reports available on an intranet by creating a symbolic link to your &HOLD& file from a local Unix based Apache web server. This article demonstrates an easy way to run Paragraphs, Sentences, and other local commands from a web browser on your intranet.

A minor configuration change to Apache and a short shell script is all that is required. We will gradually layer Unix, Web, and UniVerse access as we build towards the uv shell script. I have assumed the reader can create and edit files in Unix, and that you are working on in a non-production environment. As you will see by the end of the article, it is possible to cause damage to your UniVerse account, even though we never actually get to TCL during this exercise.

## Setting up the Web Server

As with the previous article, we assume you have an Apache webserver installed on your Unix box running UniVerse. For details on how to install Apache, and a free copy, see <http://www.apache.org>. In the previous article, we created a symbolic link to our &HOLD& file. This time, we need to make a change to the Apache **httpd.conf** configuration file to enable scripts to be run in your UniVerse home directory. I've shown part of the httpd.conf file below, with a new ScriptAlias line enabling web scripts to be executed from my /home/ross/uvhome UniVerse directory.

```
#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/usr/local/etc/apache/cgi-bin/"
ScriptAlias /uvhome/ "/home/ross/uvhome/"
```

You'll need to restart Apache with a command like:

```
kill -HUP `cat /usr/local/apache/logs/httpd.pid`
```

## Capturing Commands from the Browser at Unix

Now that we have created a pointer to the UniVerse directory, we can move there to do our work.

```
cd /home/ross/uvhome/
```

Let's create a simple script **test.cgi** to make sure we can execute commands from the web browser.

```
#!/bin/sh
```

```
echo Content-type: text/html
echo
echo Hello World
```

Make it readable and executable for everybody.

```
chmod a+rx test.cgi
```

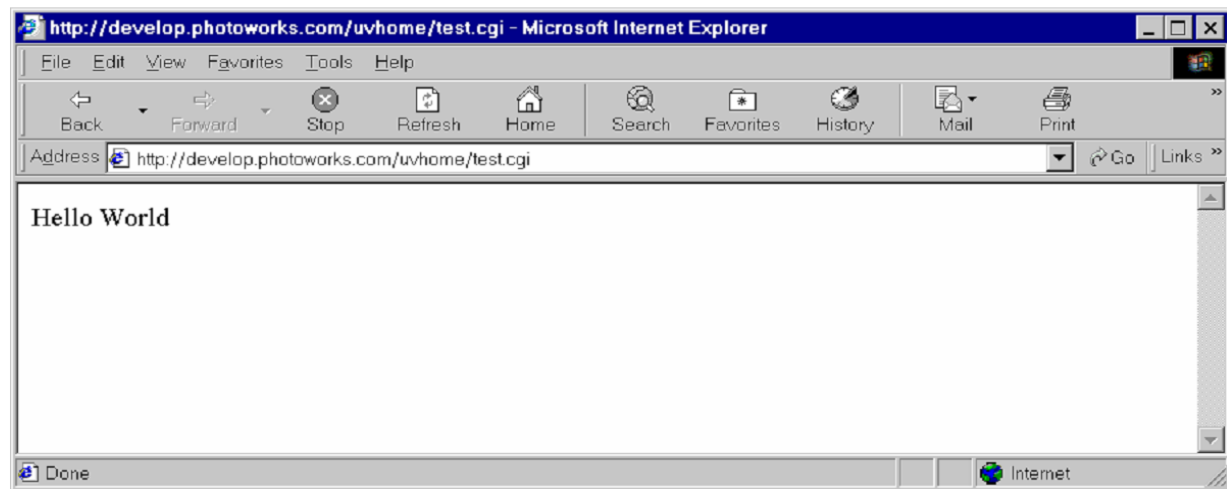
Now test it directly from Unix:

```
./test.cgi
Content-type: text/html

Hello World
```

What we see here are the two header lines that the web server needs to see in order to recognize this as valid content to return to the web browser. These header lines are not displayed to the user.

Now let's try looking at the page from a web browser...



Notice that whatever was displayed after the two header lines is passed through to the browser. Whatever we can get to display from a Unix shell can therefore be passed on to the browser, including UniVerse commands.

### Passing Commands to UniVerse from Unix

In the Unix shell in a UniVerse directory type:

```
/usr/opt/uv/bin/uv TIME
```

UniVerse starts up in this directory, executes the VOC entry for TIME, and because there is no further input required, it quits. We get a nice display of the date and time on the screen:

```
15:10:39 15 JUL 2001
```

### Passing Commands to UniVerse from a web browser

Now, rather than update our "Hello World" script to print the time, let's make it more general, able to run any UniVerse Paragraph, Sentence, or Verb. We do this by taking advantage of the web feature that allows us to perform searches and fill out forms. We can pass information through to an application by means of

command line arguments that the web server sets when it passes control to your test.cgi script. The \$\* is the entire contents of the web address following a ? after your script name, with each argument delimited by a + sign.

Let's first look at the modified script:

```
#!/bin/sh
echo Content-type: text/html
echo
/usr/opt/uv/bin/uv "$*"
```

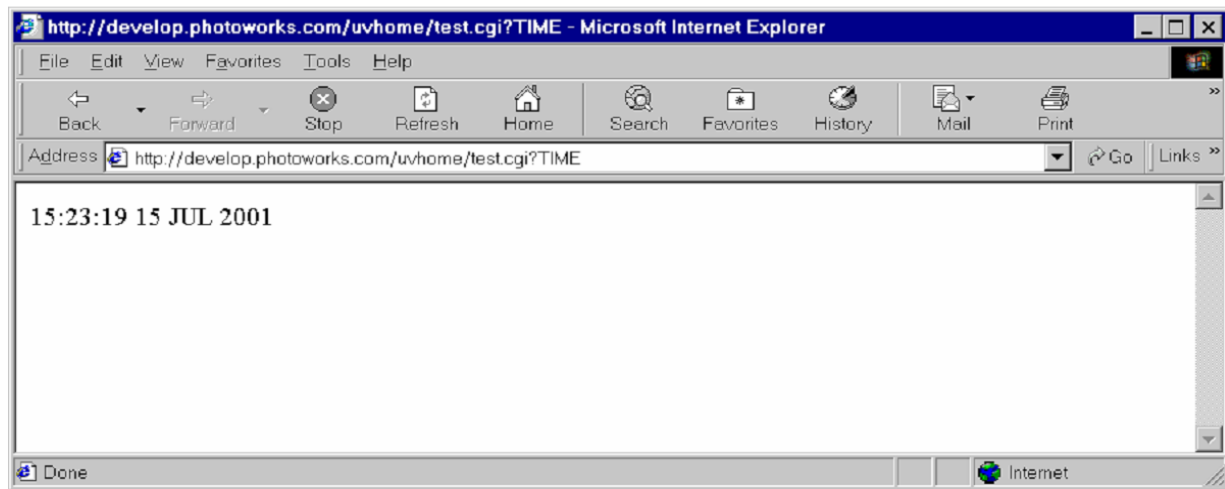
Now, let's run it at Unix, with our TIME example. Note that we are populating \$\* and making it available to our script:

```
./test.cgi TIME
Content-type: text/html

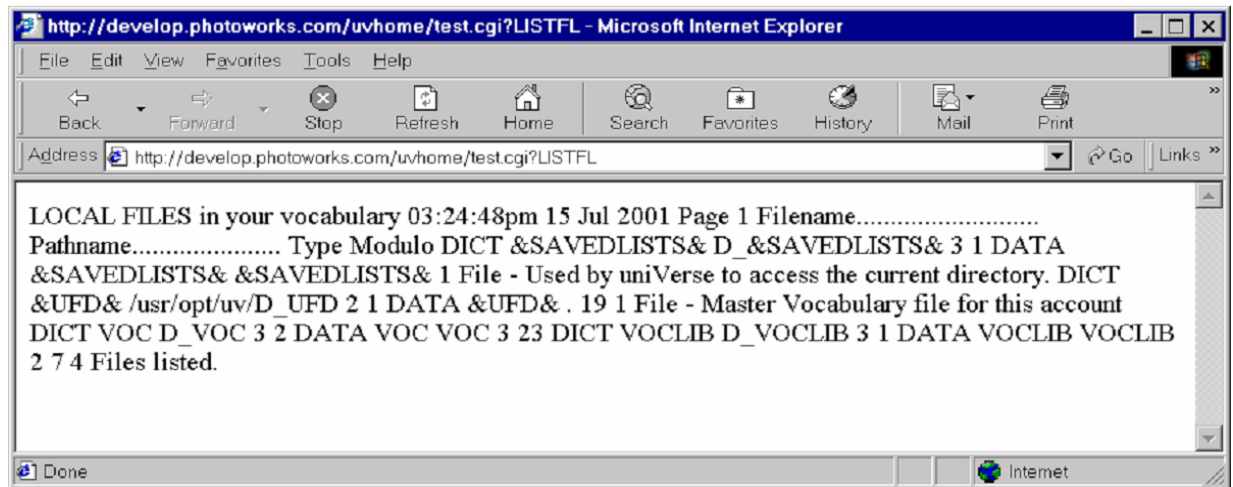
15:20:51 15 JUL 2001
```

Once again, our script prints the two header lines, then the result of our command, TIME.

Now let's look at it in a web browser:



This is a big step here, we are dynamically generating intranet content from within UniVerse, but there are some limitations. Let's try a very basic command, LISTFL:



Now this output isn't much use at all, except that we can see clearly that we are getting output from UniVerse. To make things a little clearer, lets create another script **pre.cgi**, with some html code wrapping our UniVerse call:

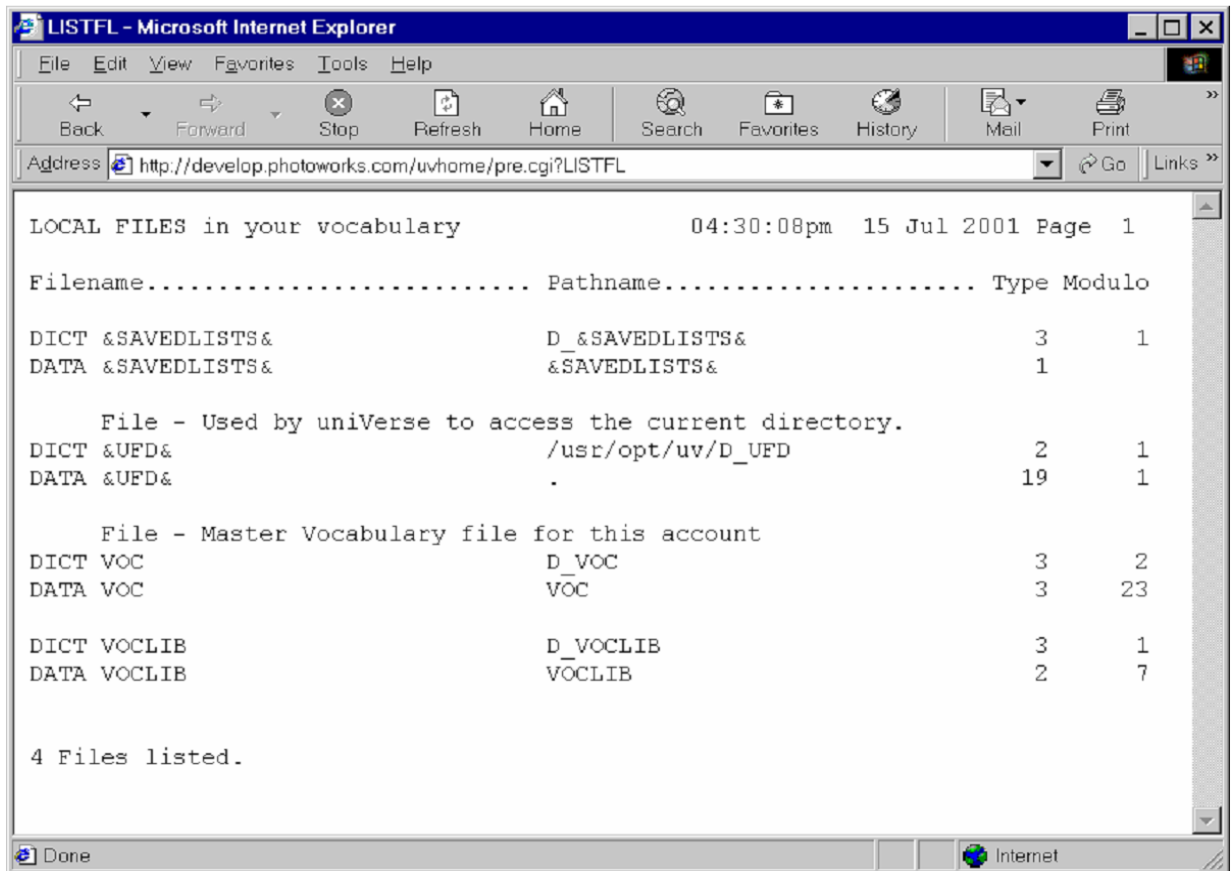
```
#!/bin/sh
echo "<html><head><title>${*}</title></head><body><pre>"
/usr/opt/uv/bin/uv "${*}"
echo "</body></html>"
```

The command being executed has been added as the title.

Make the script accessible and executable:

```
chmod a+rx pre.cgi
```

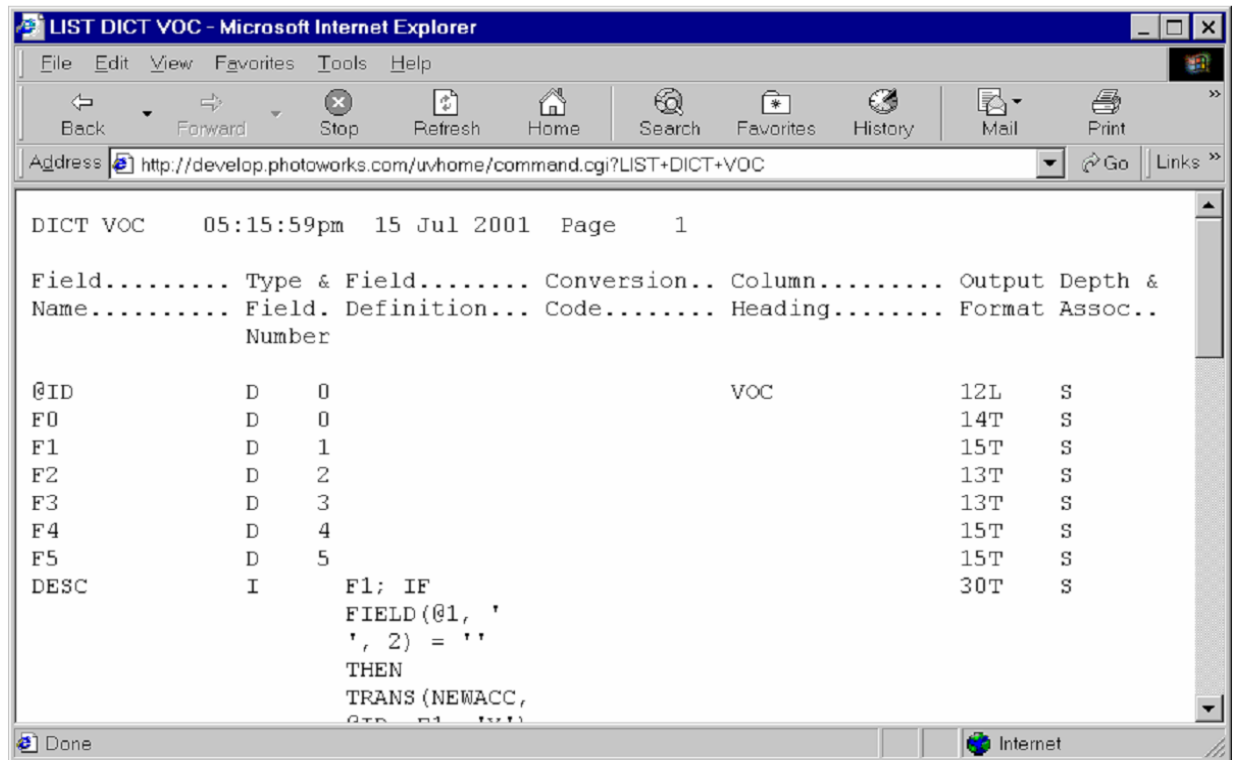
Now let's look at the output:



This is certainly more readable, and approaches usefulness. At this point you can provide output from any one word UniVerse command that would normally direct output to the screen, to a web browser.

### Passing Arguments to UniVerse

We can pass multiple word commands to the script by separating each word with a + sign in the URL. This enables us to generate any one line UniVerse command (with a few limitations involving special characters like ? and &).



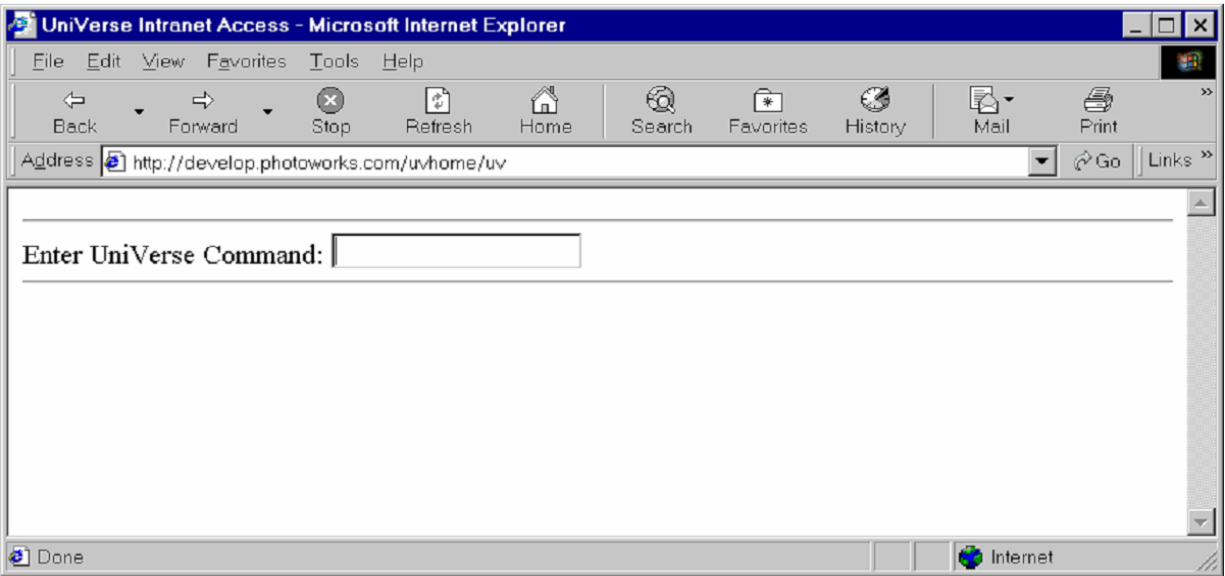
Now let's make it easy to create these multi-word commands:

Let's create our last script, **uv**, and use a classic web feature, the searchable index. It turns out that the **isindex** search creates URLs and triggers scripts in exactly the same way our script has been looking for input, with spaces swapped out for + signs.

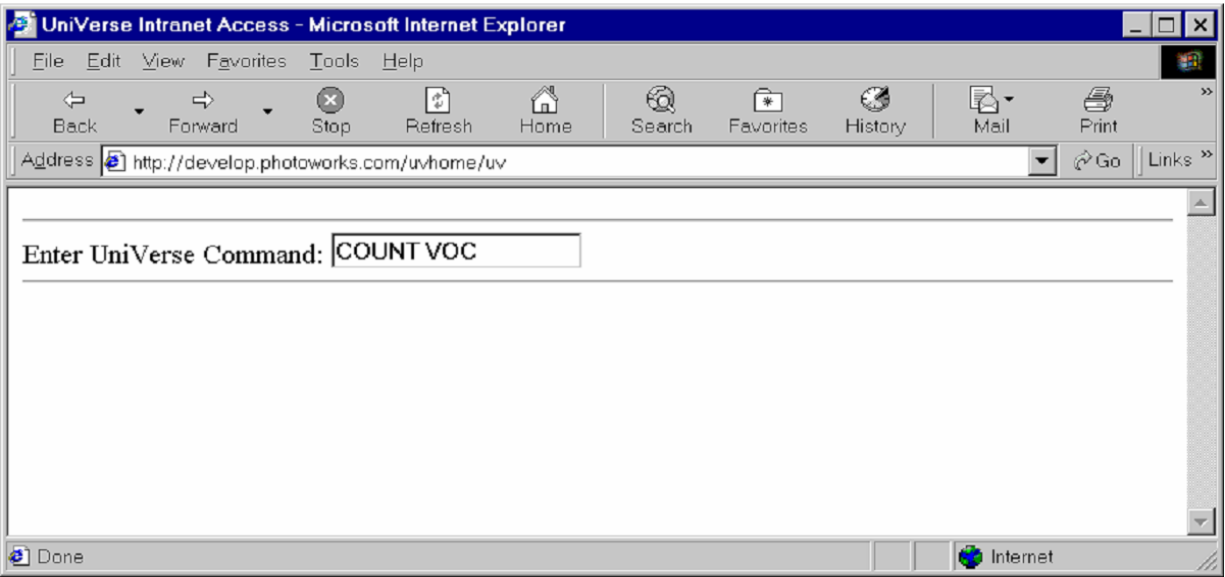
```
#!/bin/sh
echo Content-type: text/html
echo
if [ $# = 0 ]
then
  echo "<html><head><title>UniVerse Intranet Access</title>"
  echo "<isindex prompt='Enter UniVerse Command: '>"
  echo "</head></html>"
else
  echo "<html><head><title>${*}</title></head><body><pre>"
  /usr/opt/uv/bin/uv "$*"
  echo "</body></html>"
fi
```

Make this script accessible and executable:

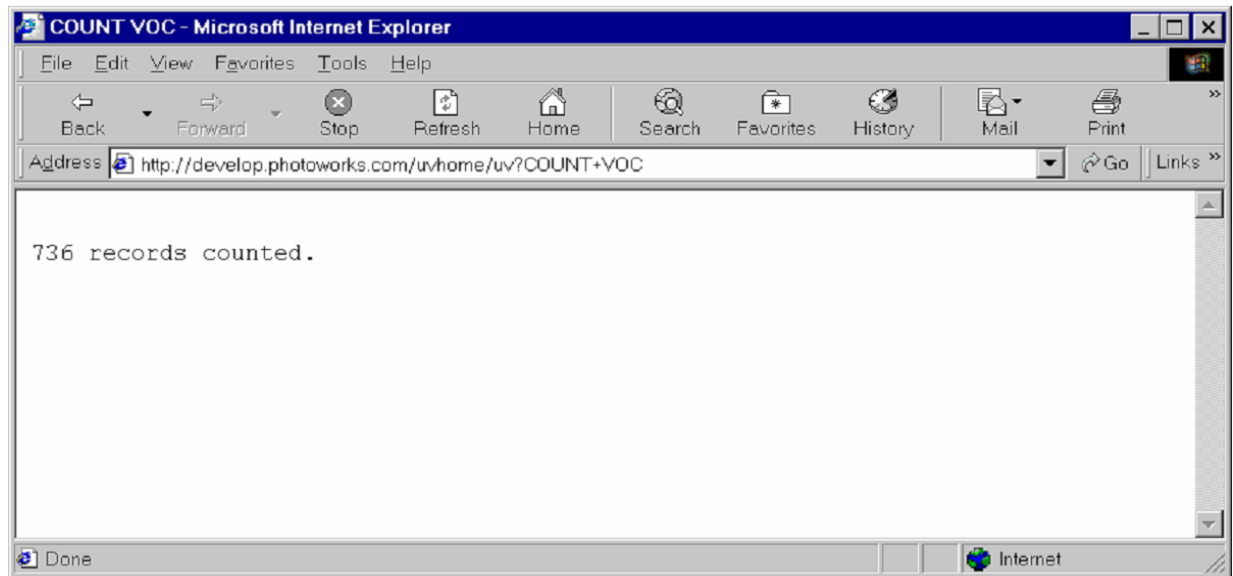
```
chmod a+rx uv
```



The first time we run the script, there are no command line arguments (if [ \$# = 0 ] is true), and the first chunk of html is displayed, including the isindex prompt.



Simply filling in a UniVerse command and hitting *[Enter]* brings up the results:



At this point, the script recognizes that there are command line arguments and executes the second chunk of html, pushing the arguments to UniVerse for execution, and returning your query results. Note that the query is displayed in the title bar. It is easy to modify the imbedded html to include the query and any other text you'd like in the body of the results.

### **Formatting your Query links**

Now that you have the ability to execute any UniVerse command from any web browser on your intranet, you can create shortcuts to common query or maintenance commands. Perhaps you have a number of UniVerse systems, and it's time consuming to log into each system to run one command - using the UniVerse Intranet Access method, you can create a web page with a series of links to each task.

### **Security**

By this time you've realized that this is very powerful. Perhaps too powerful. There is nothing stopping a user doing a DELETE.FILE VOC from this screen, with disasterous results. If your VOC is not locked down by some secure method, at a minimum I would suggest removing commands like DELETE.FILE and CLEAR.FILE from any VOC accessed in this way. It is easy to write a command preprocessor to filter queries on the UniVerse side, and my next article will start with one.

The web logs provide a detailed track of what users or workstations executed what query and when, but it's better to prevent these problems up front than to find the guilty party after the fact.

### **Limitations**

We don't have a good method of prompting the user for specific parameters for their commands or reports, and we can't access the &UFD& file - it contains special characters that cannot be passed in this form of query. These will be covered next time in Using Web Forms to access UniVerse.