Fall 2004 Handout 7

DocumentRoot, UserDir, and DirectoryIndex

Here are excerpts from the Hyperterxt Transport Protocol Dæmon configuration file **`/usr/local/apache-1.3.14/conf/httpd.conf`**. The documentation for this file is at

`http://www.apache.org/docs/mod/directives.html`

```
#If a URL specifies no filename, use this filename by default.
#If the directory has no file with this name, do an ls -l of the
#directory instead.

DirectoryIndex index.html

#In a URL, a user's loginname with a ˜ in front of it stands
#for the following subdirectory of that user's home directory.

UserDir public_html

#The following is automatically prefixed to the directory specified
#in a URL, unless the specified directory begins with a user's
#loginname with a ˜ in front of it.

DocumentRoot "/usr/local/apache-1.3.14/htdocs"

#In the URL of a gateway, the following short string stands
#for the much longer string.

ScriptAlias /cgi-bin/ "/usr/local/apache-1.3.14/cgi-bin/"
```

Uniform Resource Locators (URLs) in the World Wide Web

A *resource* is a file that can be displayed or copied, a program that can be executed, etc. For simplicity, we'll assume that the resource is a file. A *URL* is a one-line string of characters that tells your web browser how and where to get a resource for you.

Getting a resource from another host (i.e., computer on the Internet) requires the coöperation of two programs. The program running on your host is called the *client*, and the one on the host containing the resource is called the *server*. The client and server must agree on the *protocol* they will use, which is a set of rules and formats for data communication. Each protocol has a name: **`http`** (Hypertext Transport Protocol), **`ftp`** (File Transfer Protocol), etc.

The first part of a URL is the name of the protocol that the client and server will use when sending the resource from the server to the client.

`http://www.nyu.edu`	*NYU home page*
`ftp://www.nyu.edu`	*file transfer: download files from NYU*
`telnet://www.nyu.edu`	*log in to* **`www.nyu.edu`**
`ssh://www.nyu.edu`	*log in to* **`www.nyu.edu`** *(more secure)*

After the protocol comes a colon, two slashes, the name of the host that holds the file, the name of the directory that holds the file, and the name of the file.

(1) A loginname with a tilde in front of it stands for the full pathname of the **`public_html`** subdirectory of the home directory of that person. (The superuser sets this up with the **`UserDir`** in Handout 3, p. 7; it may be different on other hosts.) For example, the URL

```
http://i5.nyu.edu/~abc1234/index.html
```

refers to the file **`/home1/a/abc1234/public_html/index.html`** on the host **`i5.nyu.edu`**.

(2) If the name of the directory in the URL is not a loginname with a tilde in front of it, the prefix **`/usr/local/apache-1.3.14/htdocs`** is added to the name of the directory. (The superuser sets this up with the **`DocumentRoot`** in Handout 3, p. 7; it may be different on other hosts.) For example, the URL

```
http://i5.nyu.edu/manual/index.html
```

refers to the file **`/usr/local/apache-1.3.14/htdocs/manual/index.html`** on the host **`i5.nyu.edu`**.

(3) If the URL contains no directory at all, the directory is assumed to be **`/usr/local/apache-1.3.14/htdocs`**. (The superuser sets this up with the **`UserDir`** in Handout 3, p. 7; it may be different on other host.) For example, the URL

```
http://i5.nyu.edu/index.html
```

refers to the file **`/usr/local/apache-1.3.14/htdocs/index.html`** on the host **`i5.nyu.edu`**. (There are web pages in many languages in that directory.)

(4) If the URL contains no filename, the filename is assumed to be **`index.html`**. (The superuser sets this up with the **`DirectoryIndex`** in Handout 3, p. 7; it may be different on other hosts.) For example, the URL's

`http://i5.nyu.edu/~abc1234/` *You can omit the trailing slash.*

refer to the file **`/home1/a/abc1234/public_html/index.html`** on the host **`i5.nyu.edu`**, and the URL

`http://i5.nyu.edu/` *You can omit the trailing slash.*

refers to the file **`/usr/local/apache-1.3.14/htdocs/index.html`** on the host **`i5.nyu.edu`**.

(5) If the URL contains no filename and the directory contains no file named **`index.html`**, the browser will display an **`ls -l`** of the directory. For example, the URL

`http://i5.nyu.edu/~mm64/common/` *You can omit the trailing slash.*

will list the directory **`/home1/m/mm64/public_html/common`** on the host **`i5.nyu.edu`**.

A relative URL is like a Unix relative pathname.

(6) In a document whose own URL starts with **`http://i5.nyu.edu`**, you can omit the leading **`http://i5.nyu.edu`** from any URL. For example, you can write either of the following in your home page **`http://i5.nyu.edu/~abc1234`**:

```
http://i5.nyu.edu/~def5678/
                 /~def5678/
```

In fact, in a document whose URL starts with **`http://i5.nyu.edu/~abc1234/`**, you can omit the leading **`http://i5.nyu.edu/~abc1234/`** from any URL. For example, you can write either of the following in your home page **`http://i5.nyu.edu/~abc1234`**:

```
http://i5.nyu.edu/~abc1234/hobbies.html
                           hobbies.html
```

A URL leading to the middle of a file

(7) By default, the URL of a file leads you to the beginning of a file. You can use a **#** to make a URL leading to a point within the file. For example,

```
http://www.sandia.gov/sci_compute/elements.html#IMG
```

Here's how to do it yourself:

```
<!-- This file is ~/public_html/old_movies.html -->
<HTML>
<HEAD>
<TITLE>Old Movies</TITLE>
</HEAD>
<BODY>
<H1>Old Movies</H1>
<P>
I spend alot of time watching old movies on
<A HREF = "http://www.wnet.org">Channel 13</A>.
Blah blah blah ...
<P>
<A NAME = "favorite">
But my favorite movie is <CITE>The Wizard of Oz</CITE>.
</BODY>
</HTML>

<!-- Excerpt from your ~/public_html/index.html file. -->
<P>
I love all <A HREF = "old_movies.html">old movies</A>,
but I have my <A HREF = "old_movies.html#favorite">favorite</A>.
```

▼ **Homework 7.1: put links into your home page**

Here are links you could put into your **~/public_html/index.html** file:

```
<P>
Here are links to
<A HREF = "http://www.nyt.com/">The New York Times</A>
and
<A HREF = "http://www.ora.com/">O'Reilly Books</A>.
```

```
Here are links to The New York Times and O'Reilly Books.
```

▲

Gateways:
http://www.w3.org/CGI

A program that can be run from the World Wide Web is called a *gateway*. A gateway can be written in any language, but the textbooks usually write them in Perl.

i5 already has a directory of gateways. Some of them are written as Bourne shellscripts:

```
1$ cd /usr/local/apache-1.3.14/cgi-bin
2$ ls -l | more
-rw-r--r--   1 root     other        268 Jan 11  2001 printenv
-rw-r--r--   1 root     other        757 Jan 11  2001 test-cgi
```

```
3$ pwd
/local/apache-1.3.14/cgi-bin

4$ ls -l /usr/local/apache-1.3.14
total 368
drwxr-xr-x   2 root     other       4096 Jul  3  2002 bin
drwxr-xr-x   2 root     other       4096 Oct 28  2002 cgi-bin
-rw-r--r--   1 root     other     156160 Oct 25  2002 cgi-bin-old.tar
drwxr-xr-x   2 root     other       4096 Sep 28  2001 cgi-bin.old
drwxr-xr-x   2 root     other       4096 Oct 13 15:57 conf
drwxr-xr-x   5 root     other       4096 Feb 21  2001 htdocs
drwxr-xr-x   3 root     other       4096 Feb 21  2001 icons
drwxr-xr-x   3 root     other       4096 Jan 11  2001 include
drwxr-xr-x   2 root     other         96 Jan 11  2001 libexec
drwxr-xr-x   2 root     other         96 Nov  7 21:29 logs
drwxr-xr-x   4 root     other         96 Jan 11  2001 man
drwxr-xr-x   2 nobody   nobody        96 Jan 11  2001 proxy
```

In a URL on **i5**, the word **cgi-bin** is an abbreviation for **/usr/local/apache-1.3.14/cgi-bin**, so you can run these gateways by pointing your browser at the URL's

```
http://i5.nyu.edu/cgi-bin/printenv
http://i5.nyu.edu/cgi-bin/test-cgi
```

A World Wide Web gateway

Most of the links in a World Wide Web page lead to other pages. But a link can also lead to a program. When you click on this link, the program will run and display its output in your browser. A program run from the Web is called a *gateway*. Put your gateways in your **˜/public_html/cgi-bin** directory. The following gateway is named **classmates**.

To work correctly, a gateway must obey a set of rules called the *Common Gateway Interface*, or CGI. See **http://www.w3.org/CGI/**. For example, the first line of a gateway's standard output must begin with **Content-type:**. The **C** must be uppercase; there must be a dash, not an underscore; there must be no space on either side of the dash; and there must be a colon, not a semicolon. The second line of standard output must be empty. If the gateway runs a program without redirecting the program's standard output and standard error output (for example, **echo** and **comm** in the following gateway), then these outputs will be displayed in the web browser.

Since your gateway may be run by people other than you, it cannot use any of the variables created in your **.profile** file, e.g., your **$PATH**. In your gateway you must therefore write the full pathname of any command which is in an out-of-the-way directory, e.g. your personal **bin** or **cgi-bin** subdirectories. Or in your gateway you can say

```
echo '<P>'
echo My original '$PATH' was $PATH
export PATH=/home1/a/abc1234/bin:/home1/a/abc1234/public_html/cgi-bin:$PATH
echo '<BR>'
echo My new '$PATH' is $PATH
```

Until now, you have put all your executable files in your **˜/bin** directory. But your gateway must go in your **˜/public_html/cgi-bin** directory.

```
#!/bin/ksh
#Gateway script to output the loginnames of the classmates logged in.

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Classmates logged in</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Classmates logged in</H1>'
echo '<PRE>'

~mm64/bin/roster 46 | sort > /tmp/inclass$$
who | awk '{print $1}' | sort | uniq > /tmp/loggedin$$
comm -12 /tmp/inclass$$ /tmp/loggedin$$
rm /tmp/inclass$$ /tmp/loggedin$$

echo '</PRE>'
echo '</BODY>'
echo '</HTML>'
exit 0
```

`chmod` your gateway **`classmates`** to **`rwxr-xr-x`** so that everyone in the world can execute it. In Homeworks 1.2 and 1.3 you already **`chmod`**'ed your home, **`public_html`**, and **`cgi-bin`** directories to **`rwxr-xr-x`**.

When testing your gateway, be sure to execute **`~/public_html/cgi-bin/classmates`**, not **`~/bin/classmates`**:

```
1$ cd ~/public_html/cgi-bin
2$ pwd

3$ classmates | more                              the wrong shellscript
4$ ~/public_html/cgi-bin/classmates | more
5$ ./classmates | more      Dot means your current directory, Handout 1, p. 9.
```

The gateway should produce the following output:

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Classmates logged in</TITLE>
</HEAD>
<BODY>
<H1>Classmates logged in</H1>
<PRE>
abc1234
def5678
ghi0912
</PRE>
</BODY>
</HTML>
```

Write the following in your **~/public_html/index.html** file:

```
<P>
Click
<A HREF = "http://i5.nyu.edu/cgi-bin/cgiwrap/abc1234/classmates">here</A>
to see the classmates logged in right now.
```

Because of the line

```
ScriptAlias /cgi-bin/ "/usr/local/apache-1.3.14/cgi-bin/"
```

in the file **/usr/local/apache-1.3.14/conf/httpd.conf**, the link actually runs the gateway **/usr/local/apache-1.3.14/cgi-bin/cgiwrap**. **cgiwrap** receives an environment variable named **$PATH_INFO** containing the string **abc1234/classmates**, which makes **cgiwrap** run your gateway **/home1/a/abc1234/public_html/cgi-bin/classmates**.

Better yet, all you need is a relative URL:

```
<P>
Click
<A HREF = "/cgi-bin/cgiwrap/abc1234/classmates">here</A>
to see the classmates logged in right now.
```

```
Click here to see the classmates logged in right now.
```

When you click here, the browser will render the output of the gateway as

```
Classmates logged in

abc1234
def5678
ghi9012
```

▼ Homework 7.2: write a gateway

Write a gateway that doesn't always produce exactly the same output each time you run it. It can't be the gateway shown above that outputs the loginnames of the people in the class who are logged in now. It could output

(1) the current date and time;

(2) a calendar of the current month;

(3) a picture of the current phase of the moon (Handout 2, p. 14, line 93);

(4) the number of people logged into i5.nyu.edu;

(5) the gateway's PID number;

(6) the names and contents of all the gateway's environment variables. Use the **env** program in Handout 1, p. 3, line 46, and Handout 3, p. 14.

(7) the contents of just one environment variable, e.g. the gateway's **$PATH** variable. Also see Mark's gateway **http://i5.nyu.edu/cgi-bin/cgiwrap/mm64/whatismyip**

(8) whether or not you're logged into i5.nyu.edu (see "Where is Mark logged into i5.nyu.edu right now?" in his home page, and note that only a live human being, not a gateway, can give the lowercase **-m** option to **who**. A gateway would have to get the same effect by filtering the standard output of **who** through **grep**.);

(9) etc.

The gateway should not attempt to perform input—we'll do that later. Do only output. If you're outputting a picture (e.g., rows and columns of numbers and/or words, or a picture of the moon), don't forget to enclose it in the **<PRE>** and **</PRE>** in Handout 3, p. 9, lines 63 and 68.

Hand in a printout of your **index.html** file, your gateway, and the standard output of your gateway. Please circle the link in your home page that leads to your gateway.
▲

Insert HTML tags into the output of a gateway

```
#!/bin/ksh
#Gateway script to output the loginnames of the classmates logged in.

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Classmates logged in</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Classmates logged in</H1>'
echo '<OL>'

~mm64/bin/roster 46 | sort > /tmp/inclass$$
who | awk '{print $1}' | sort | uniq > /tmp/loggedin$$
comm -12 /tmp/inclass$$ /tmp/loggedin$$ | sed 's/^/<LI>/'
rm /tmp/inclass$$ /tmp/loggedin$$

echo '</OL>'
echo '</BODY>'
echo '</HTML>'
exit 0
```

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Classmates logged in</TITLE>
</HEAD>
<BODY>
<H1>Classmates logged in</H1>
<OL>
<LI>abc1234
<LI>def5678
<LI>ghi0912
</OL>
</BODY>
</HTML>
```

```
Classmates logged in

1. abc1234
2. def5678
3. ghi9012
```

Insert links into the output of a gateway

See X52.9545 Handout 8, p. 7 (or pp. 323−324 in the textbook) for **&** in the replacement part of an **s** command.

```
#!/bin/ksh
#Gateway script to output the loginnames of the classmates logged in.

echo Content-type: text/html
echo
echo ’<HTML>’
echo ’<HEAD>’
echo ’<TITLE>Classmates logged in</TITLE>’
echo ’</HEAD>’
echo ’<BODY>’
echo ’<H1>Classmates logged in</H1>’
echo "Click on a loginname to go to that person’s home page."
echo ’<OL>’

~mm64/bin/roster 46 | sort > /tmp/inclass$$
who | awk ’{print $1}’ | sort | uniq > /tmp/loggedin$$
comm -12 /tmp/inclass$$ /tmp/loggedin$$ |
    sed ’s:.*:<LI><A HREF = "/~&/">&</A>:’
rm /tmp/inclass$$ /tmp/loggedin$$

echo ’</OL>’
echo ’</BODY>’
echo ’</HTML>’
exit 0
```

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Classmates logged in</TITLE>
</HEAD>
<BODY>
<H1>Classmates logged in</H1>
Click on a loginname to go to that person’s home page.
<OL>
<LI><A HREF = "/~abc1234/">abc1234</A>
<LI><A HREF = "/~def5678/">def5678</A>
<LI><A HREF = "/~ghi9012/">ghi9012</A>
</OL>
</BODY>
</HTML>
```

```
Classmates logged in

Click on a loginname to go to that person’s home page.
1. abc1234
2. def5678
3. ghi9012
```

▼ Homework 7.3: write a gateway with sed

Write a gateway that doesn't always produce exactly the same output each time you run it. It can't be the gateway shown above that outputs the loginnames of the people in the class who are logged in now. Put a **`sed`** in the gateway (as in the above examples) to insert HTML tags into the output of the gateway.

Hand in a printout of your **`index.html`** file, your gateway, and the standard output of your gateway. Please circle the link in your home page that leads to your gateway.

▲

Other types of Content-type

See the file **`/usr/local/etc/httpd/conf/mime.types`** for the complete list of **`Content-type`**'s.

```
#!/bin/sh

echo Content-type: text/plain
echo

date
echo
cal

exit 0
```

```
#!/bin/sh
echo Content-type: image/gif
echo

cat /home1/m/mm64/public_html/markface.gif
exit 0
```

```
#!/bin/sh
echo Content-type: image/gif
echo

giftoppm /home1/m/mm64/public_html/markface.gif |
/home/m/mm64/46/ppm/bin/ppm_negative |
ppmtogif

exit 0
```

Here's one of the above gateways in C:

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    FILE *fp = fopen("/home1/m/mm64/public_html/markface.gif", "r");
    int c;

    if (fp == NULL) {
        printf ("Content-type: text/plain\n");
        printf ("\n");
```

```
            printf ("Can’t open /home1/m/mm64/public_html/markface.gif\n");
            exit (EXIT_FAILURE);
        }

        printf ("Content-type: image/gif\n");
        printf ("\n");

        while ((c = getc(fp)) != EOF) {
            putchar(c);
        }

        fclose (fp);
        exit (EXIT_SUCCESS);
    }
```

Of course, you can combine

```
printf ("Content-type: image/gif\n");
printf ("\n");
```

to

```
printf ("Content-type: image/gif\n\n");
```

```
#!/bin/sh
echo Location: http://i5.nyu.edu/˜mm64
echo

exit 0
```

```
#!/bin/sh
#This gateway is /home1/a/public_html/cgi-bin/showthem.
#Go to abc1234’s home page if they have one.  Otherwise,
#merely display their line in the /etc/passwd file.

homedir=‘awk -F: ’$1 == "abc1234" {print $6}’ /etc/passwd‘

if [ -f $homedir/public_html/index.html -a \
    -r $homedir/public_html/index.html ]
then
    echo Location: http://i5.nyu.edu/˜abc1234
    echo
else
    echo Content-type: text/plain
    echo
    grep abc1234 /etc/passwd
fi

exit 0
```

Another example of **`Location:`** is **`$mer/public_html/cgi-bin/classphoto`**.

```
#!/bin/sh
#Copied from /usr/local/apache-1.3.14/cgi-bin/donothing.

echo Status: 204 Do Nothing
echo

mail abc1234 < /home1/a/letter

exit 0
```

Environment variables

The program **env** outputs the names and values of your environment variables.

```
#!/bin/sh
#This program is /home1/a/abc1234/public_html/cgi-bin/myprog.

echo Content-type: text/plain
echo

env

echo
echo My loginname is $LOGNAME
echo My loginname is ‘whoami‘
exit 0
```

```
DOCUMENT_ROOT=/usr/local/etc/httpd/htdocs
GATEWAY_INTERFACE=CGI/1.1
HTTP_ACCEPT=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
HTTP_ACCEPT_CHARSET=iso-8859-1,*,utf-8
HTTP_ACCEPT_LANGUAGE=en
HTTP_CONNECTION=Keep-Alive
HTTP_HOST=i5.nyu.edu
HTTP_REFERER=http://i5.nyu.edu/cgi-bin/cgiwrap/˜mm64/myprog
HTTP_USER_AGENT=Mozilla/4.03 (Macintosh; U; PPC, Nav)
PATH=/home/r/rosenblg/bin:/usr/local/bin:/usr/sbin:/usr/local/etc:
    /usr/ucb:/usr/bin:/bin:/usr/lbin:/etc:/usr/bin/X11:/sbin:
    /pmdf/com:.:/usr/local/gnat/bin
PATH_INFO=
PATH_TRANSLATED=/home1/m/mm64/public_html/cgi-bin/myprog
QUERY_STRING=
REMOTE_ADDR=192.168.20.107
REMOTE_HOST=ed132-m2.ed.acf.nyu.edu
REMOTE_PORT=1123
REQUEST_METHOD=GET
REQUEST_URI=/cgi-bin/cgiwrap/˜mm64/myprog
SCRIPT_FILENAME=/usr/local/etc/httpd/cgi-bin/cgiwrap
SCRIPT_NAME=/cgi-bin/cgiwrap/mm64/myprog
SERVER_ADMIN=webmaster@nyu.edu
SERVER_NAME=i5.nyu.edu
SERVER_PORT=80
SERVER_PROTOCOL=HTTP/1.0
SERVER_SOFTWARE=Apache/1.2.4
```

```
My loginname is
My loginname is abc1234
```

& is a special character in the second half of an **s** command in **sed** and **vi**.

```
#!/bin/sh

echo 'Content-type: text/html'
echo

echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Environment variables</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Environment variables</H1>'
echo '<PRE>'
env | sed '
    s/&/\&amp;/g
    s/</\&lt;/g
    s/>/\&gt;/g
'
echo '</PRE>'
echo '</BODY>'
echo '</HTML>'

exit 0
```

Command line arguments

In a URL, write a question mark before the first command line argument of a gateway, and a plus sign before each subsequent argument:

```
http://i5.nyu.edu/cgi-bin/cgiwrap/~abc1234/printargs?arg1+arg2+arg3
```

```
#!/bin/sh
#This gateway is $HOME/public_html/cgi-bin/printargs

echo 'Content-type: text/plain'
echo

for arg
do
    echo "'"$arg"'"
done

exit 0
```

Your browser will display

```
'arg1'
'arg2'
'arg3'
```

Hex escape sequences

A URL cannot contain a blank, so instead of writing

```
http://i5.nyu.edu/cgi-bin/cgiwrap/~abc1234/printargs?first arg+second arg
```

you must write

http://i5.nyu.edu/cgi-bin/cgiwrap/˜abc1234/printargs?first%20arg+second%20arg

Your browser will display

```
'first arg'
'second arg'
```

Use the same trick if one of the arguments must contain a plus sign. See **man 5 ascii | grops | lpr**.

▼ Homework 7.4: add a command line argument

Let the loginname be passed as a command line argument to the above gateway **showthem**. Then you can say

```
<P>
Click on each person to see their home page if they have one,
or their line in the /etc/passwd file if they don't.
<OL>
<LI><A HREF = /cgi-bin/cgiwrap/˜abc1234/showthem?def5678">def5678</A>
<LI><A HREF = /cgi-bin/cgiwrap/˜abc1234/showthem?ghi9012">ghi9012</A>
<LI><A HREF = /cgi-bin/cgiwrap/˜abc1234/showthem?jkl3456">jkl3456</A>
</OL>
```

▲

A web page that contains a form

A *form* is a part of a World Wide Web page that contains buttons, checkboxes, places to fill-in words, etc. Here are the URL's of pages containing a form:

http://i5.nyu.edu/˜mm64/cal.html	*the example below*
http://i5.nyu.edu/˜mm64/man/	*Unix* **man** *page*
http://i5.nyu.edu/˜mm64/x52.9232/	*C program to output a color flag*
http://i5.nyu.edu/˜mm64/x52.9264/	*C++ program to output a color flag*
http://www.usps.gov/ncsc/lookups/lookup_zip+4.html	*nine-digit zip codes*

The following file is **˜/public_html/cal.html**. Its URL is therefore **http://i5.nyu.edu/˜abc1234/cal.html**. It should have the same nine permission bits (**rw-r--r--**) as your web home page **˜/public_html/index.html**.

The **INPUT** widgets must be between the **FORM** tags in lines 8 and 45. To show the user where the form begins and ends, surround it with horizontal rules in lines 7 and 46. When you press the **SUBMIT** button in line 42, the gateway in line 8 will be run.

```
 1 <HTML>
 2 <HEAD>
 3 <TITLE>Calendar form</TITLE>
 4 </HEAD>
 5 <BODY>
 6
 7 <HR>
 8 <FORM METHOD = POST ACTION = "http://i5.nyu.edu/cgi-bin/cgiwrap/mm64/cal">
 9 <H2>Calendar form</H2>
10
11 <P>
12 This form runs the gateway
13 <CODE>/home1/m/mm64/public_html/cgi-bin/cal</CODE>
```

```
14 on the host
15 <CODE>i5.nyu.edu</CODE>.
16 <P>
17 Which year do you want to see?
18 <INPUT TYPE = TEXT NAME = "year" SIZE = 5>
19
20 <P>
21 Which month do you want to see?
22 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 1>January
23 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 2>February
24 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 3>March
25 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 4>April
26 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 5>May
27 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 6>June
28 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 7>July
29 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 8>August
30 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 9>September
31 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 10>October
32 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 11 CHECKED>November
33 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 11>November
34 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = 12>December
35 <BR><INPUT TYPE = RADIO NAME = "month" VALUE = "">All twelve months
36
37 <P>
38 <INPUT TYPE = CHECKBOX NAME = "date">Check here
39 to see the current date and time too.
40
41 <P>
42 <INPUT TYPE = SUBMIT VALUE = "See the calendar.">
43 <BR>
44 <INPUT TYPE = RESET VALUE = "Start again.">
45 </FORM>
46 <HR>
47
48 </BODY>
49 </HTML>
```

To make a pop-up menu, change the above lines 22−35 to

```
50 <SELECT NAME = "month">
51 <OPTION VALUE = 1>January
52 <OPTION VALUE = 2>February
53 <OPTION VALUE = 3>March
54 <OPTION VALUE = 4>April
55 <OPTION VALUE = 5>May
56 <OPTION VALUE = 6>June
57 <OPTION VALUE = 7>July
58 <OPTION VALUE = 8>August
59 <OPTION VALUE = 9>September
60 <OPTION VALUE = 10>October
61 <OPTION VALUE = 11 SELECTED>November
62 <OPTION VALUE = 12>December
63 <OPTION VALUE = "">All 12 months
64 </SELECT>
```

We’ll start with a simple gateway that merely displays the data it received from the above form.

The environment variable **$CONTENT_LENGTH** tells the gateway the number of bytes it should read the standard input. These bytes constitute one long line, without an end-of-file or even a newline character (**\n**) at the end:

```
year=2004&month=1&date=on
```

In C,

```
#include <stdio.h>
#include <stdlib.h>

    char *const p = getenv("CONTENT_LENGTH");
    int content_length;

    if (p == NULL) {
        printf("Content-type: text/plain\n");
        printf("\n");
        printf("CONTENT_LENGTH variable is missing.\n");
        exit (EXIT_FAILURE);
    }

    content_length = atoi(p);
```

The command **head -25** inputs 25 lines from the standard input and outputs them to the standard output. The command **/usr/local/bin/head -c25** inputs 25 bytes (“characters”) from the standard input and outputs them to the standard output. See Handout 4, p. 27. The **/usr/local/bin/head** command in the following gateway therefore inputs **$CONTENT_LENGTH** bytes from the gateway’s standard input and outputs them to the standard output, which is displayed in the browser.

```
1$ man head                     documentation about /bin/head
2$ man -M /usr/local/man head   documentation about /usr/local/bin/head
```

```
#!/bin/ksh
#This gateway is ~/public_html/cgi-bin/cal.

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Calendar</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Calendar</H1>'
echo The first $CONTENT_LENGTH bytes of standard input are
echo '<PRE>'
/usr/local/bin/head -c$CONTENT_LENGTH
echo '</PRE>'
echo '</BODY>'
echo '</HTML>'
exit 0
```

Here is the output of the above gateway if the user checks the **CHECKBOX**. If the user doesn’t check it, the **&date=on** would not be there.

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Calendar</TITLE>
</HEAD>
<BODY>
<H1>Calendar</H1>
The first 25 bytes of standard input are
<PRE>
year=2004&month=1&date=on</PRE>
</BODY>
</HTML>
```

It appears in your browser like this:

```
Calendar

The first 25 bytes of standard input are
year=2004&month=1&date=on
```

Pipe the standard output of **head** into **tr**, which chops the long line of bytes into several shorter lines. See the **tr** in Handout 3, p. 13.

```
#!/bin/ksh
#This gateway is ~/public_html/cgi-bin/cal.

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Calendar</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Calendar</H1>'
echo The first $CONTENT_LENGTH bytes of standard input are
echo '<PRE>'
#Change ampersands to newlines.
/usr/local/bin/head -c$CONTENT_LENGTH | tr '&' '\012'
echo                                #Output a final newline.
echo '</PRE>'
echo '</BODY>'
echo '</HTML>'
exit 0
```

Here is the output of the above gateway:

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Calendar</TITLE>
</HEAD>
<BODY>
<H1>Calendar</H1>
The first 25 bytes of standard input are
<PRE>
year=2004
month=1
date=on
</PRE>
</BODY>
</HTML>
```

It appears in your browser like this:

```
Calendar

The first 25 bytes of standard input are
year=2004
month=1
date=on
```

Now that we’ve verified that the gateway is receiving data from the form, let’s make the gateway chop the data up and store it in shell variables. Could you use the parentheses from the double extra credit part of Homework 3.9 to create the file **`/tmp/$$`** more simply?

```
#!/bin/ksh
#This gateway is ˜/public_html/cgi-bin/cal.
#It displays a calendar and is executed from the form in the page
#˜/public_html/cal.html
#The URL of that page is
#http://i5.nyu.edu/˜mm64/cal.html

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Calendar</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Calendar</H1>'
echo '<PRE>'

/usr/local/bin/head -c$CONTENT_LENGTH | tr '&' '\012' > /tmp/$$
echo >> /tmp/$$         #append a newline to the end of the temp file

 year=`awk -F= '$1 == "year"  {print $2}' /tmp/$$`
month=`awk -F= '$1 == "month" {print $2}' /tmp/$$`
 date=`awk -F= '$1 == "date"  {print $2}' /tmp/$$`
rm /tmp/$$

#Quotes make sure that cal will receive a second argument even if
#$year is the null string.  In the absence of a second argument,
#cal would think that the first (and only) argument was the year.

cal $month "$year"
echo '</PRE>'

#Need quotes because $date would be the null string if checkbox not
#checked.
if [[ "$date" == on ]]
then
    echo '<P>'
    date
fi

echo '</BODY>'
echo '</HTML>'
exit 0
```

The **`/tmp/$$`** file contains three lines:

```
year=2004
month=1
date=on
```

Here is the output of the above gateway:

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Calendar</TITLE>
</HEAD>
<BODY>
<H1>Calendar</H1>
<PRE>
   January 2004
 S  M Tu  W Th  F  S
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

</PRE>
<P>
Fri Jan  9 00:41:39 EST 2004
</BODY>
</HTML>
```

It is rendered in your browser like this:

```
Calendar

   January 2004
 S  M Tu  W Th  F  S
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31


Fri Jan  9 00:41:39 EST 2004
```

▼ Homework 7.5: write a form that gives input to a gateway

Write a form that gives input to a gateway that doesn't always produce the same output each time you run it. It can't be the gateway and form shown above.

Hand in a printout of the **`.html`** file that contains the form, your gateway shellscript, and (if possible) snapshots of the form and the output of the gateway.

▲

It's easier to get data from a form in Perl

`require` in Perl is like **`#include`** in C and C++. **`cgi-lib.pl`** is in the directory **`/usr/local/lib/perl5`** on i5, and its home page is **`http://www.bio.cam.ac.uk/cgi-lib`**. **`ReadParse`** is one of the functions in this library.

`$|` flushes the buffer after each output.

We're giving a comma-separated list of three strings to the **`print`** statement:

```
    print ’hello’, ’goodbye’, ’hello again’;
    print ’date’, ‘date‘;
```

Note that a quoted string can go on, and on, and on.

```
#!/usr/local/bin/perl

$| = 1;
require ’cgi-lib.pl’;
ReadParse();

print ’Content-type: text/html

<HTML>
<HEAD>
<TITLE>Calendar</TITLE>
</HEAD>
<BODY>
<H1>Calendar</H1>
<PRE>
’,
    ‘cal $in{month} $in{year}‘,
    ’</PRE>
</BODY>
</HTML>
’;

exit 0;
```

□