Lecture 22

# 408/508 *Computational Techniques for Linguists*

# Today's Topic

Sending form information to a Webserver program.

- Two methods:
  1. GET method
     - last time: `<form>` with `action="program.cgi"`, `method="GET"`
  2. POST method
     - **Homework 8**

# Worked example: `CMUDict`

- **Note**: the Python file (I had in `~/Sites`):
  - `cmudictpy.cgi` could also be placed in `/Library/WebServer/CGI-Executables` (macOS)
- **Then** client request:
  - `<form action=http://localhost/cgi-bin/cmudictpy.cgi" method="GET">`

  instead of:
  - `<form action="http://localhost/~user/cmudictpy.cgi" method="GET">`
- Webserver runs the program:
  - `/Library/WebServer/CGI-Executables/cmudictpy.cgi (macOS)`
  - `/var/www/html/cmudictpy.cgi (Ubuntu)`

# Sending information using **GET**

First: Last: Submit

- HTML form:

```
1. <form action="http://localhost/cgi-bin/get.cgi" method="GET">
2. First: <input type="text" name="first" size=12>
3. Last:  <input type="text" name="last" size=12>
4. <input type="submit">
5. </form>
```

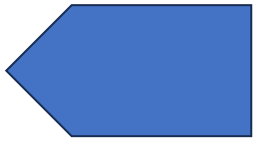
```
http://localhost/cgi-bin/get.cgi?first=Sandiway&last=Fong
```

| Character | URL Encoded |
|---|---|
| ; | %3B |
| ? | %3F |
| / | %2F |
| : | %3A |
| # | %23 |
| & | %26 |
| = | %3D |
| + | %2B |
| $ | %24 |
| , | %2C |
| <space> | %20 or + |
| % | %25 |
| < | %3C |
| > | %3E |
| ~ | %7E |
| % | %25 |

- Information encoded using alphanumeric characters: why?
- because URLs are restricted to alphanumeric characters only
- **bash** accesses the URL-encoded string via the environment variable **`QUERY_STRING`**

# Today's Topic

- Sending form information to a Webserver program.
- Two methods:
  1. GET method
  2. **POST method**



# Sending information using **POST**




- File: `form-post.html` could be in `/Library/WebServer/Documents`
- URL: `http:/localhost/form-post.html`
- HTML form:

```
1. <form action="http://localhost/cgi-bin/read.cgi" method="POST">
2. First: <input type="text" name="first" size=12>
3. Last:  <input type="text" name="last" size=12>
4. <input type="submit">
5. </form>
```

- **bash** accesses the URL-encoded string via command `read`
  - cf. GET using **`QUERY_STRING`**

# Sending information using **POST**

- **bash** accesses the URL-encoded string on standard input via read

- read.cgi in /Library/WebServer/CGI-Executables:

```
1.#!/bin/bash
2.echo "Content-Type: text/plain"
3.echo
4.read input
5.origIFS=$IFS
6.IFS='=&'
7.set -- $input
8.IFS=$origIFS
9.echo "Form data \$2:<$2> \$4:<$4>"
```

**read**

Read a line from standard input

```
Syntax
      read [-ers] [-a aname] [-p prompt] [-t timeout]
              [-n nchars] [-d delim] [name...]
```

IFS = **internal field separator** (for arguments)
default: space newline tab
set it to = and & because
first=String1&last=String2

**set -- *String***
-- option: positional parameters **$1,$2**,..etc. are set after splitting *String*

# Sending information using **POST**

- Client-side code:

```
1 <!DOCTYPE HTML>¶
2 <html>¶
3   <head>¶
4     <title>CGI POST Example</title>¶
5   </head>¶
6   <body>¶
7     <h1>CGI POST Example</h1>¶
8     <form action="http://localhost/cgi-bin/read.cgi" method="POST">
9       First: <input type="text" name="first" size=12>¶
10      Last:  <input type="text" name="last" size=12>¶
11      <input type="submit">¶
12    </form>¶
13  </body>¶
14 </html>¶
```

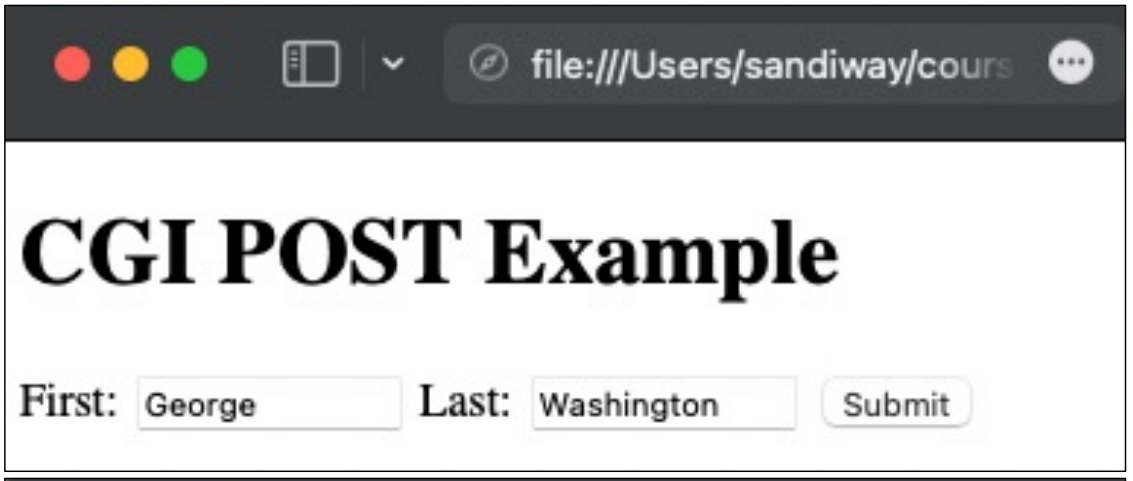
File `form-post.html` place in
`macOS: /Library/Webserver/Documents/`
`Ubuntu: /var/www/html`

Server-side code:

```
1. #!/bin/bash
2. echo "Content-Type: text/plain"
3. echo
4. read input
5. origIFS=$IFS
6. IFS='=&'
7. set -- $input
8. IFS=$origIFS
9. echo "Form data \$2:<$2> \$4:<$4>"
```

File `read.cgi` place in
`/Library/Webserver/CGI-Executables/`
Ubuntu: `/usr/lib/cgi-bin`

Sending information using POST

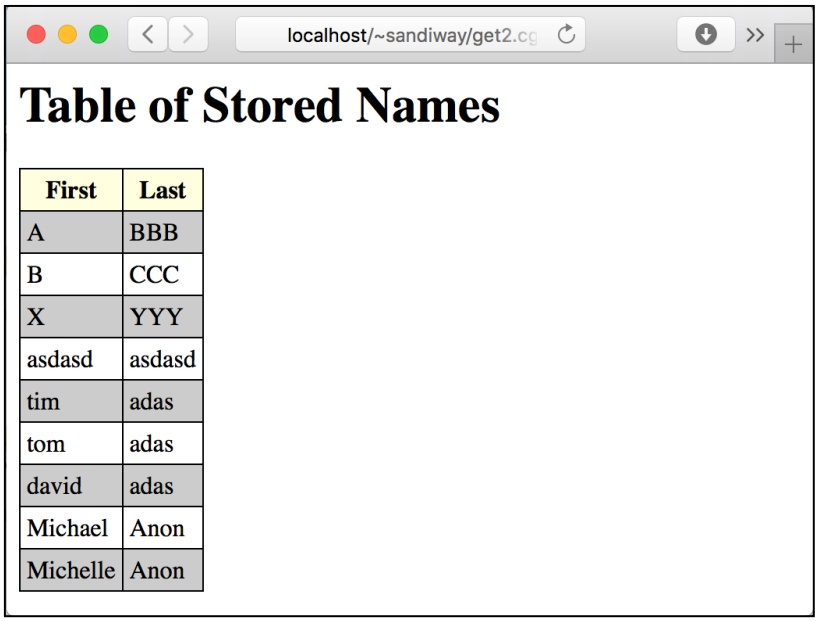



```
first=George&last=Washington
$1    $2     $3   $4
```

# Example: adding names to a list on a server

- *Normally, you'd set up relational database software,e.g. mysql, on the webserver*




| First | Last |
|---|---|
| A | BBB |
| B | CCC |
| X | YYY |
| asdasd | asdasd |
| tim | adas |
| tom | adas |
| david | adas |
| Michael | Anon |
| Michelle | Anon |

# Example: adding names to a list on a server

**Client side:**
`addnames.html`

**Server side:**
`get2.cgi`
`names.txt` (database of names)

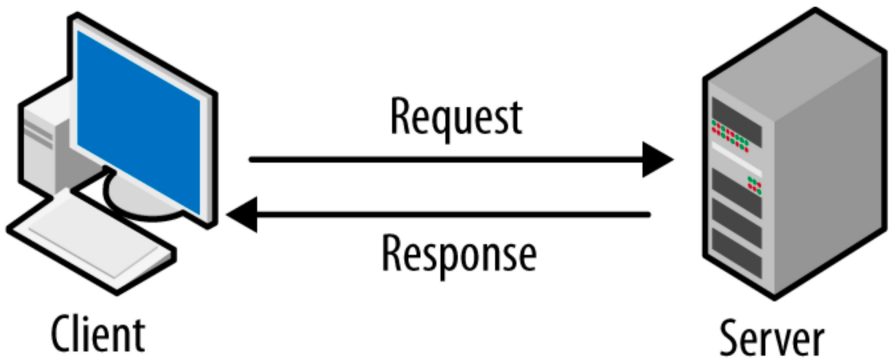



image from https://madooei.github.io/cs421_sp20_homepage/client-server-app/

# A Note on Permissions

- **Caution!**
  - the Webserver must be able to write to the directory in which `names.txt` is stored (*to be able to update it*)
  - this is not you (as user), but typically the user is `_www` or `www`.
- Suppose we run it in the user directory `~/Sites` (`~/public_html` Ubuntu).
- On macOS:
  - `~/Sites` must have world write permission
  - `chmod a+w ~/Sites`
- On Ubuntu:
  - `~/public_html` must have world write permission
  - `chmod a+w ~/public_html`

# Example: adding names to a list on a server

## Server-side: `get2.cgi`

```
#!/bin/bash
echo "Content-Type: text/html; charset=utf-8"
echo
echo "<html><head><style>"
echo "table { border-collapse: collapse }"
echo "td, th { border: 1px solid; padding: 4px }"
echo "th { background-color: lightyellow }"
echo "tr:nth-child(even) { background-color: #ccc }"
echo "</style></head>"
echo "<body><h1>Table of Stored Names</h1>"
echo "<form action="http://localhost/~sandiway/get2.cgi" method="GET">"
echo "First: <input type="text" name="first" size=12>"
echo "Last:  <input type="text" name="last" size=12>"
echo "<input type="submit"></form>"
origIFS=$IFS
IFS='=&'
set -- $QUERY_STRING
IFS=$origIFS
# names.txt must exist and
# be write-able for everyone (www)!
echo "$2 $4" >> names.txt
echo "<table><tr><th>First</th><th>Last</th></tr>"
while read -r first last
do
    echo "<tr><td>$first</td><td>$last</td></tr>"
done < names.txt
echo "</table>"
echo "</body></html>"
exit 0
```

## Client-side: `addnames.html`

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>CGI GET Example</title>
 </head>
 <body>
  <h1>CGI GET Example</h1>
  Adds names to a list kept on the webserver.
  <br>
  Prints out the saved list.
  <p></p>
  <form action="http://localhost/~sandiway/get2.cgi" method="GET">
   First: <input type="text" name="first" size=12>
   Last:  <input type="text" name="last" size=12>
   <input type="submit">
  </form>
 </body>
</html>
```

**Create database file**: in `~/Sites`

```
touch names.txt      (create an empty file)
chmod a+w names.txt
```

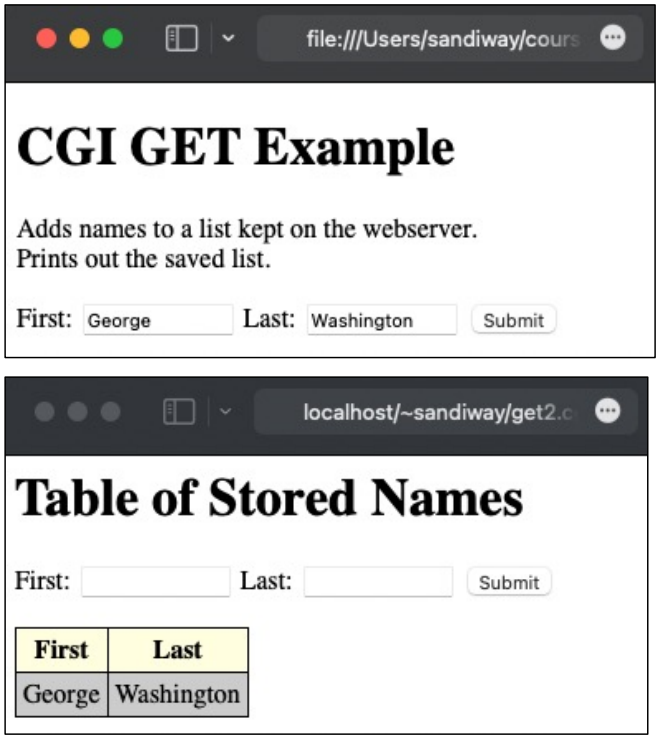
# Example: adding names to a list on a server

- **Terminal** (in ~/Sites):

```
Sites$ touch names.txt
Sites$ chmod a+w names.txt
Sites$ ls -l names.txt
-rw-rw-rw-  1 sandiway  staff  0
Nov  8 19:46 names.txt
```

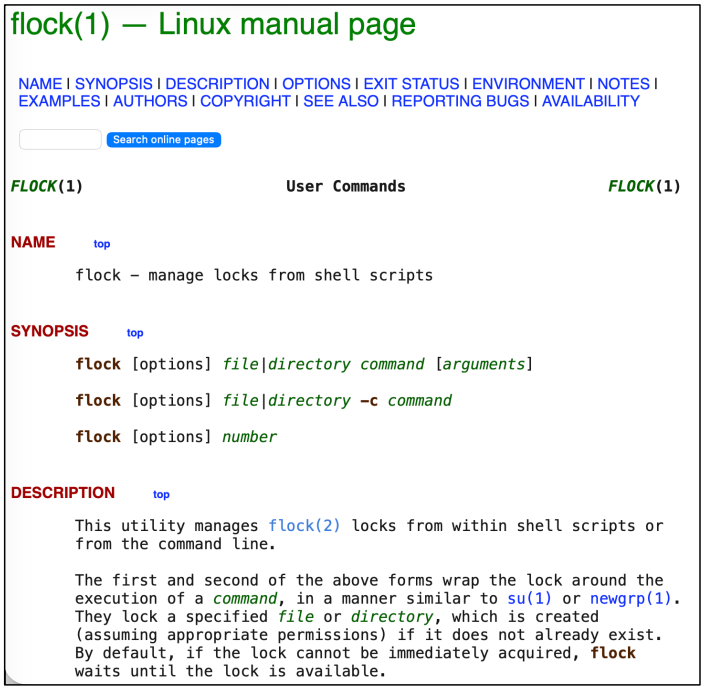
- After **Submit** is pressed:

```
Sites$ ls -l names.txt
-rw-rw-rw-  1 sandiway  staff  18
Nov  8 19:49 names.txt
(base) Sites$ cat names.txt
George Washington
```



# Example: adding names to a list on a server

- There is a potential problem with simultaneous updates
  - *Let's see if we can simulate it!*
- Normally, we'd use a **mutex** mechanism, e.g.
- file locking: `flock`



flock(1) — Linux manual page

NAME | SYNOPSIS | DESCRIPTION | OPTIONS | EXIT STATUS | ENVIRONMENT | NOTES | EXAMPLES | AUTHORS | COPYRIGHT | SEE ALSO | REPORTING BUGS | AVAILABILITY

Search online pages

**FLOCK**(1) User Commands **FLOCK**(1)

**NAME** top

flock - manage locks from shell scripts

**SYNOPSIS** top

**flock** [options] *file*|*directory command* [*arguments*]

**flock** [options] *file*|*directory* **-c** *command*

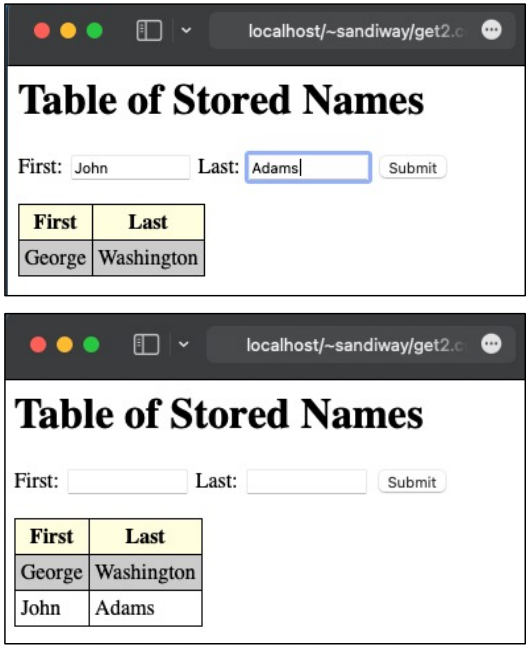
**flock** [options] *number*

**DESCRIPTION** top

This utility manages flock(2) locks from within shell scripts or from the command line.

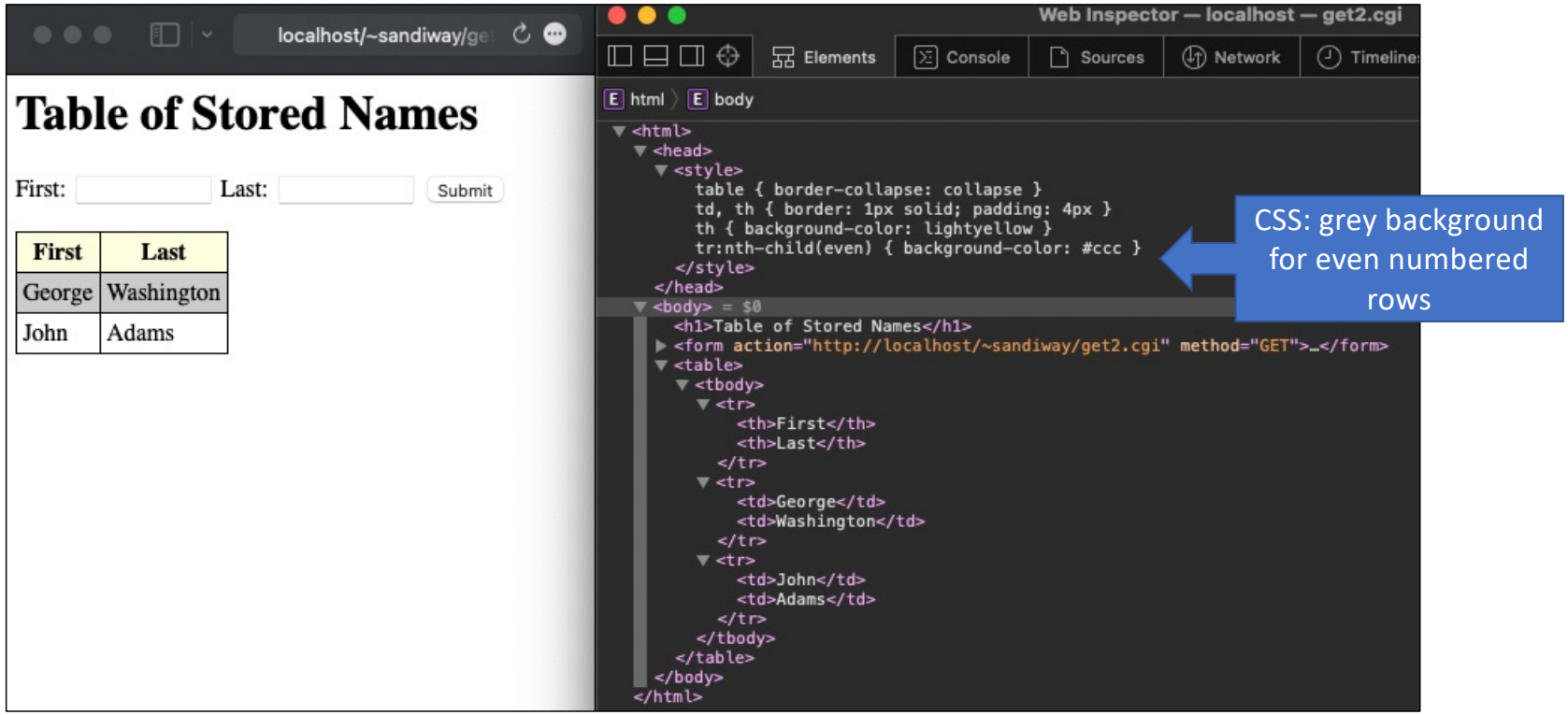
The first and second of the above forms wrap the lock around the execution of a *command*, in a manner similar to su(1) or newgrp(1). They lock a specified *file* or *directory*, which is created (assuming appropriate permissions) if it does not already exist. By default, if the lock cannot be immediately acquired, **flock** waits until the lock is available.

# Example: adding names to a list on a server




```
Sites$ ls -l names.txt
-rw-rw-rw-  1 sandiway  staff  29 Nov  8 19:53 names.txt
Sites$ cat names.txt
George Washington
John Adams
```

# Example: adding names to a list on a server



# Example: adding/deleting names on a server

**Table of Stored Names**

First: Last: Add: Submit

| First | Last | |
|---|---|---|
| First | Last | Delete |
| John | A | Delete |
| Mary | B | Delete |

**Client side:**
`addnames3.html`

**Server side:**
`get3.cgi`
`names.txt`

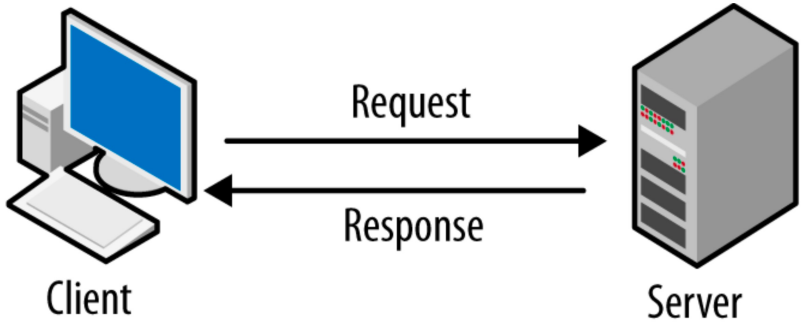
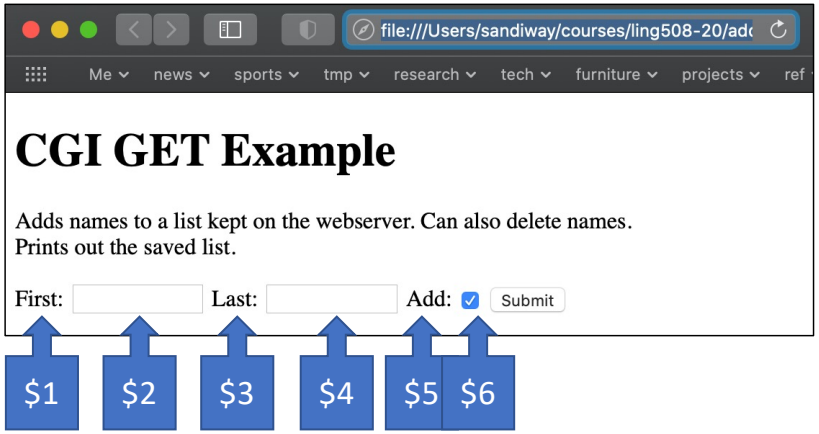



image from https://madooei.github.io/cs421_sp20_homepage/client-server-app/

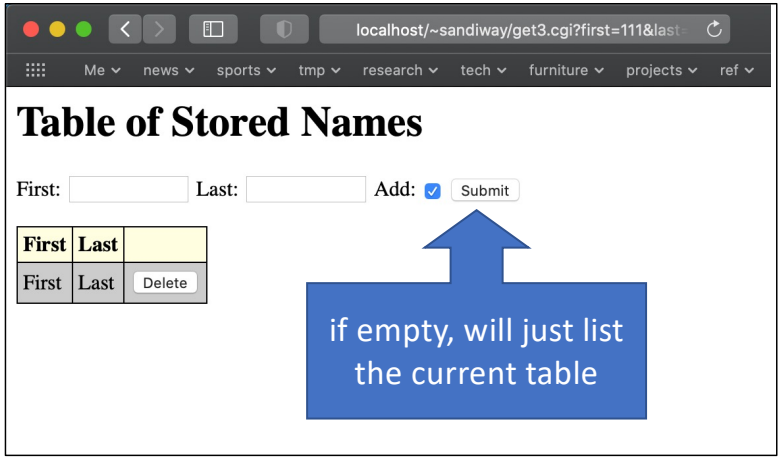
# addnames3.html

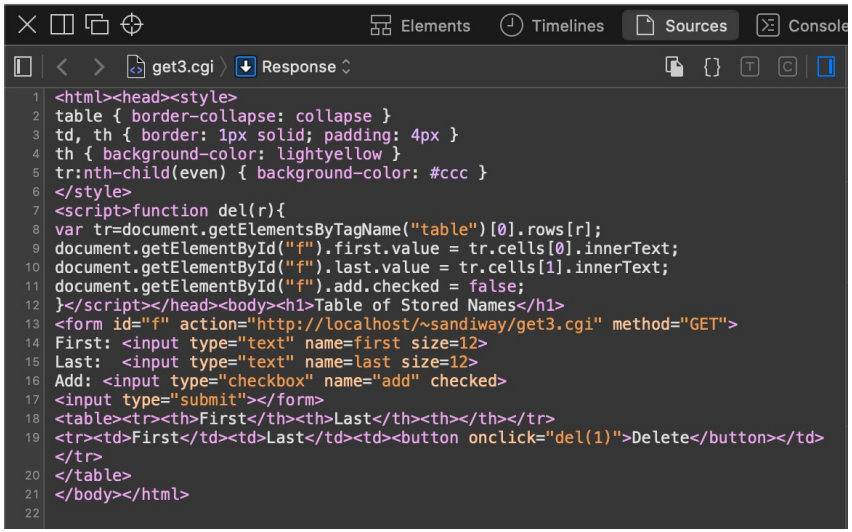
```
1 <!DOCTYPE HTML>¶
2 <html>¶
3   <head>¶
4     <title>CGI GET Example</title>¶
5   </head>¶
6   <body>¶
7     <h1>CGI GET Example</h1>¶
8     Adds names to a list kept on the webserver. Can also delete names.¶
9     <br>¶
10    Prints out the saved list.¶
11    <p></p>¶
12    <form action="http://localhost/~sandiway/get3.cgi" method="GET">  ¶
13      First: <input type="text" name="first" size=12>¶
14      Last:  <input type="text" name="last" size=12>¶
15      Add: <input type="checkbox" name="add" checked>¶
16      <input type="submit">¶
17    </form>¶
18  </body>¶
19 </html>¶
```




**Note**: actually $1, $3 and $5 are names "first", "last" and "add"

# get3.cgi



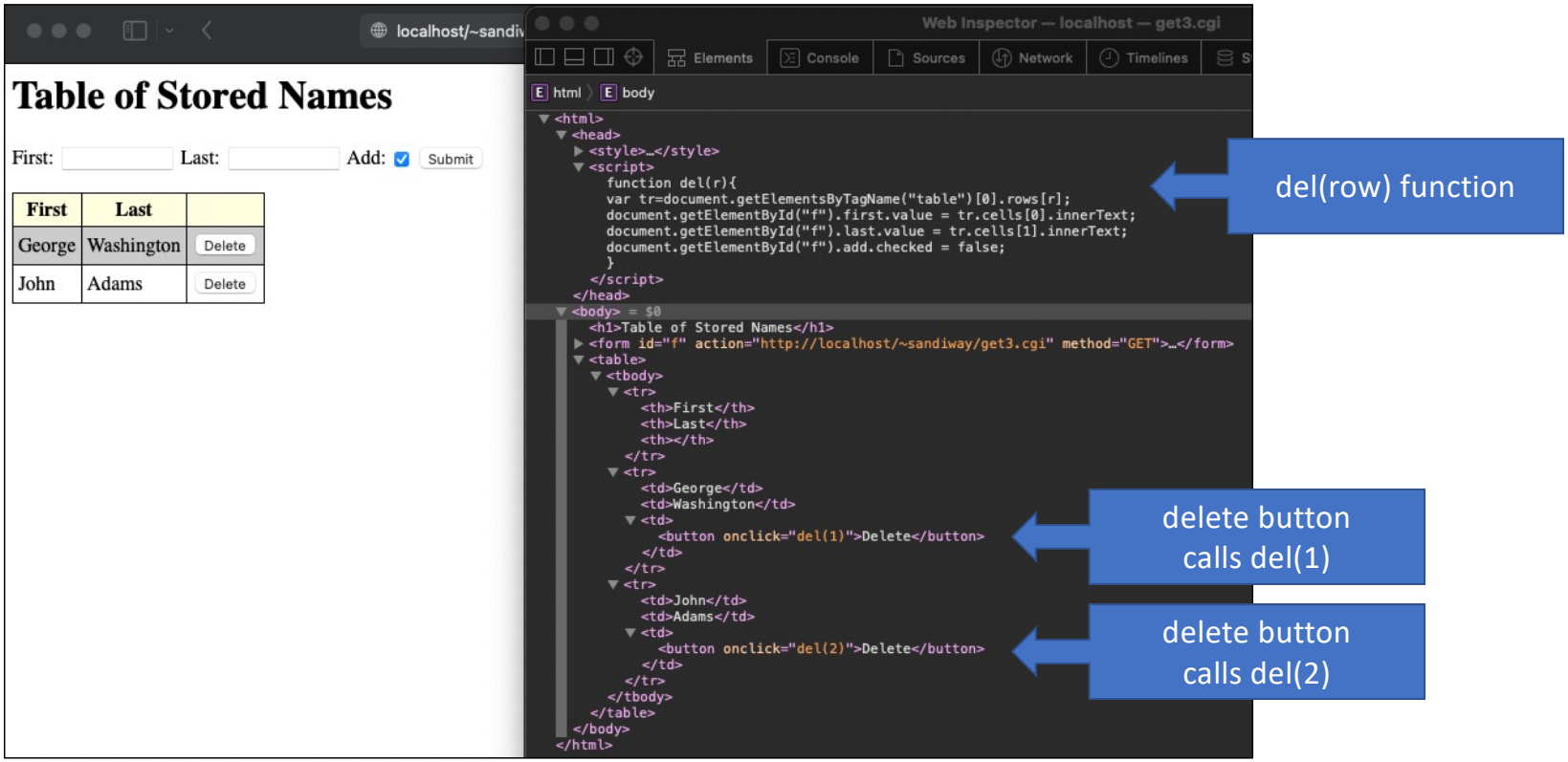

Note: action is `http://localhost/~sandiway/get3.cgi`

# get3.cgi

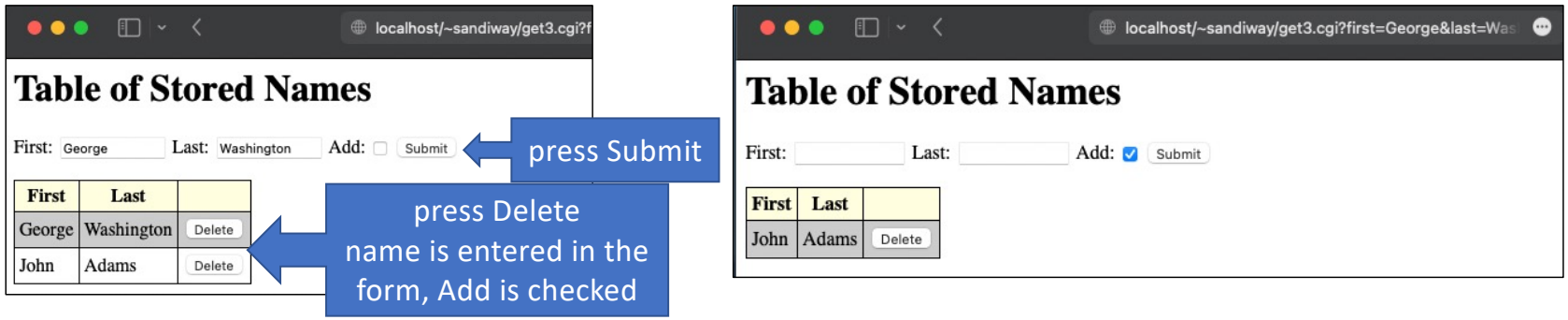



- Current state of names.txt

# get3.cgi

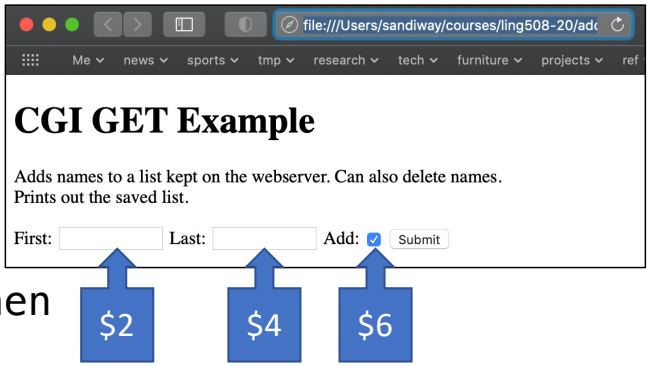

localhost/~sandi
Table of Stored Names
First: Last: Add: Submit
First Last
George Washington Delete
John Adams Delete
Web Inspector — localhost — get3.cgi
Elements Console Sources Network Timelines
html body
<html>
<head>
<style>…</style>
<script>
function del(r){
var tr=document.getElementsByTagName("table")[0].rows[r];
document.getElementById("f").first.value = tr.cells[0].innerText;
document.getElementById("f").last.value = tr.cells[1].innerText;
document.getElementById("f").add.checked = false;
}
</script>
</head>
<body> = $0
<h1>Table of Stored Names</h1>
<form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">…</form>
<table>
<tbody>
<tr>
<th>First</th>
<th>Last</th>
<th></th>
</tr>
<tr>
<td>George</td>
<td>Washington</td>
<td>
<button onclick="del(1)">Delete</button>
</td>
</tr>
<tr>
<td>John</td>
<td>Adams</td>
<td>
<button onclick="del(2)">Delete</button>
</td>
</tr>
</tbody>
</table>
</body>
</html>
del(row) function
delete button calls del(1)
delete button calls del(2)

# get3.cgi




http://localhost/~sandiway/get3.cgi?first=George&last=Washington

# get3.cgi




```
if [ "$6" = 'on' ]; then
    if [ "$2" != '' ] && [ "$4" != '' ]; then
      echo "$2 $4" >> names.txt
    fi
else
    perl -i -ne "print unless m/^$2 $4$/" names.txt
fi
```

**Using the -i Option** `perl -i`

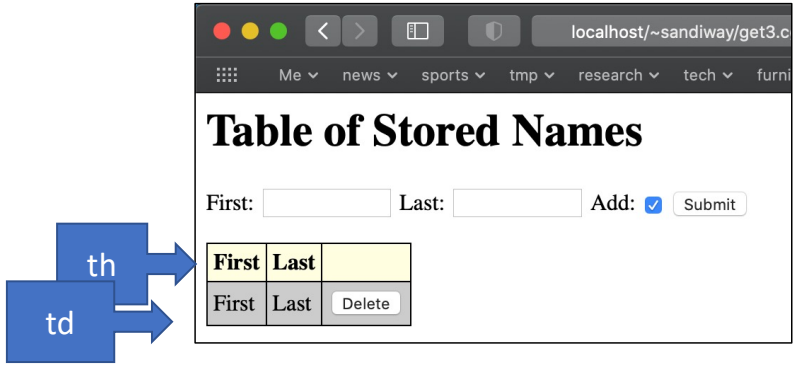
The `-i` option lets you modify files in-place. This means that Perl will automatically rename the input file and open the output file using the original name. You can force Perl to create a backup file by specifying a file extension for the backup file immediately after the `-i`. For example, `-i.bak`. If no extension is specified, no backup file will be kept.

# get3.cgi

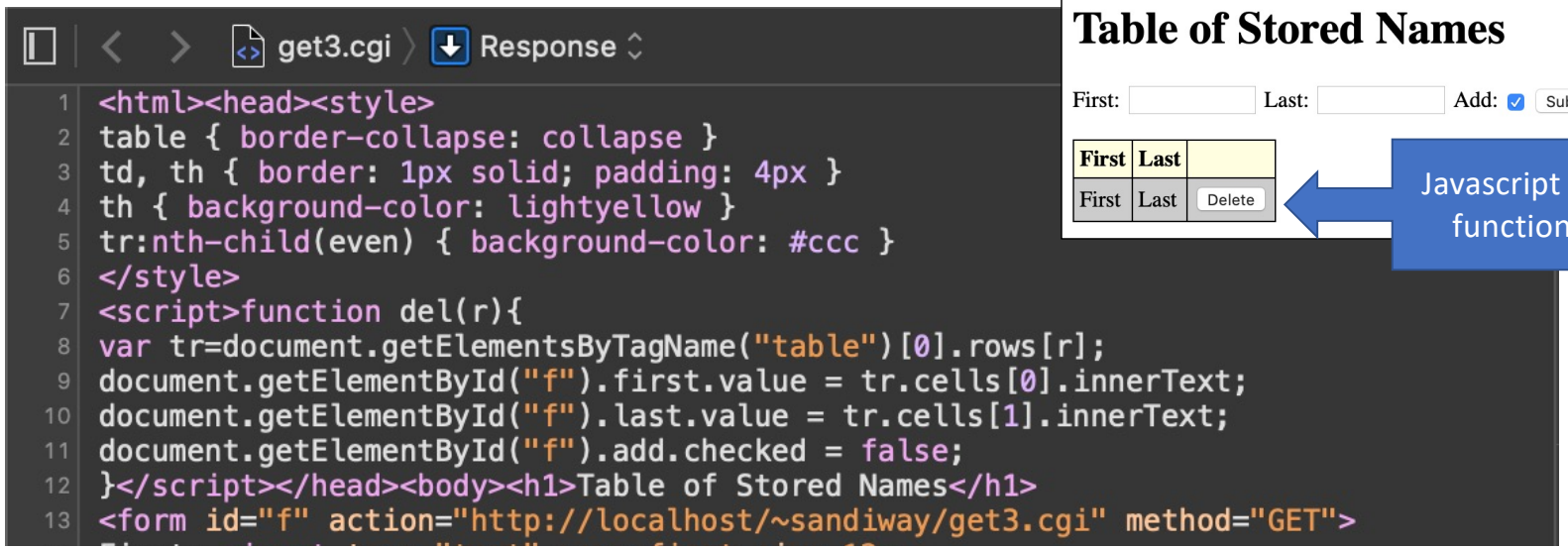
```
echo "<table><tr><th>First</th><th>Last</th><th></th></tr>"
i=1
while read -r first last
do
    echo "<tr><td>$first</td><td>$last</td><td><button
onclick=\"del($i)\">Delete</button></td></tr>"
    ((i++))
done < names.txt
echo "</table>"
echo "</body></html>"
```

`-r` do not allow backslashes to escape any characters

`< names.txt` means read from file names.txt



# get3.cgi


get3.cgi
Response
1 <html><head><style>
2 table { border-collapse: collapse }
3 td, th { border: 1px solid; padding: 4px }
4 th { background-color: lightyellow }
5 tr:nth-child(even) { background-color: #ccc }
6 </style>
7 <script>function del(r){
8 var tr=document.getElementsByTagName("table")[0].rows[r];
9 document.getElementById("f").first.value = tr.cells[0].innerText;
10 document.getElementById("f").last.value = tr.cells[1].innerText;
11 document.getElementById("f").add.checked = false;
12 }</script></head><body><h1>Table of Stored Names</h1>
13 <form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">
localhost/~sandiway/get3.c
Me news sports tmp research tech furni
Table of Stored Names
First: Last: Add: Submit
First Last
First Last Delete
Javascript call to function del
First: First Last: Last Add: Submit
first.value
last.value
checked = false

# Homework 8

1. Configure your Apache2 webserver to permit `cgi-bin` executable files (*see previous lectures*)
2. Adapt the name database files (*modify the content*) and `cgi-bin` files
3. Show it working (e.g. add/delete Pokémon characters)
4. Take screenshots!

- Usual rules:
  - one PDF file
  - Subject: 408/508 Homework 8 YOUR NAME
  - due date: midnight Sunday