Lecture 18

408/508 *Computational Techniques for Linguists*

Today's Topics

- Homework 7 graded.
- Homework 8 – setting up `httpd` to serve:
  - http://localhost/ (*system-wide homepage*)
  - http://localhost/~yourusername/ (*your personal homepage*)
- Today we discuss running programs on the webserver itself.
  - in the default `cgi-bin` directory
  - two example: one `text/plain`, one `text/html`
  - inside your home directory (*next time*)

Homework 8

- For Mac and Ubuntu users
  - set up Apache2 in your system: *bear in mind the different file locations*
- In each case:
  - http://localhost/ (*system-wide homepage*)
  - http://localhost/~yourusername/ (*your personal homepage*)
  - Create two different `index.html` webpages at these locations, e.g. add your photo on your user homepage and your computer's photo on the system-wide homepage
  - Show your system works! (*include screen snapshots please*)
  - Submit one PDF file (by Sunday midnight)

CGI-bin

- https://en.wikipedia.org/wiki/Common_Gateway_Interface

> In computing, **Common Gateway Interface** (**CGI**) is an interface specification for web servers to execute programs like console applications (also called command-line interface programs) running on a server that generates web pages dynamically. Such programs are known as *CGI scripts* or simply as *CGIs*. The specifics of how the script is executed by the server are determined by the server. In the common case, a CGI script executes at the time a request is made and generates HTML.[1]

- Again, Apple and Linux are different …

test.cgi

- On course webpage:

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
echo "Running a Bash shell script from cgi-bin successfully!"
echo -n "Now it's: "
date
echo -n "User: "
whoami
echo -n "Directory: "
ls -l
exit 0
```

tells the receiving browser what to expect: output of each command is sent to the browser

Boilerplate browser expects is:

```
Content-Type: text/plain
<blank line>
```

Content

date

NAME

date – display or set date and time

DESCRIPTION

When invoked without arguments, the date utility displays the current date and time.

whoami

NAME

whoami – display effective user id

SYNOPSIS

whoami

DESCRIPTION

The whoami utility has been obsoleted by the id(1) utility, and is equivalent to “id -un”. The command “id -p” is suggested for normal interactive use.

The whoami utility displays your effective user ID as a name.

`cgi-bin` on macOS

```
1#!/bin/bash¶
2echo "Content-Type: text/plain"¶
3echo¶
4echo "Running a Bash shell script from cgi-bin successfully!"¶
5echo -n "Now it's: "¶
6date¶
7echo -n "User: "¶
8whoami¶
9echo -n "Directory: "¶
10ls -l¶
11exit 0¶
```

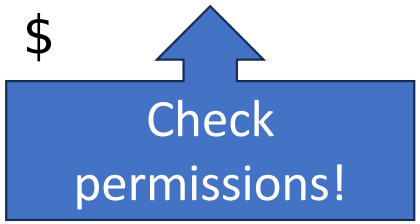
- Put it in directory
  - `/Library/WebServer/CGI-Executables`
  - `sudo` needed (owned by *root*)
- URL:
  - http://localhost/cgi-bin/test.cgi

test.cgi must be executable and readable chmod 755

WRONG! *Prints the contents of the script instead of running it*!

`cgi-bin` on macOS

```
$ cd /Library/WebServer/CGI-Executables/
$ ls -l test.cgi
-rwxr-xr-x  1 root  wheel  196 Nov  1  2022 test.cgi
$
```



`cgi-bin` on macOS

- Assuming the webserver is running,

```
$ ps -ax | grep httpd
17614 ??         0:09.45 /usr/sbin/httpd -D FOREGROUND
17618 ??         0:00.00 /usr/sbin/httpd -D FOREGROUND
17619 ??         0:00.01 /usr/sbin/httpd -D FOREGROUND
26312 ttys000    0:00.00 grep httpd
```

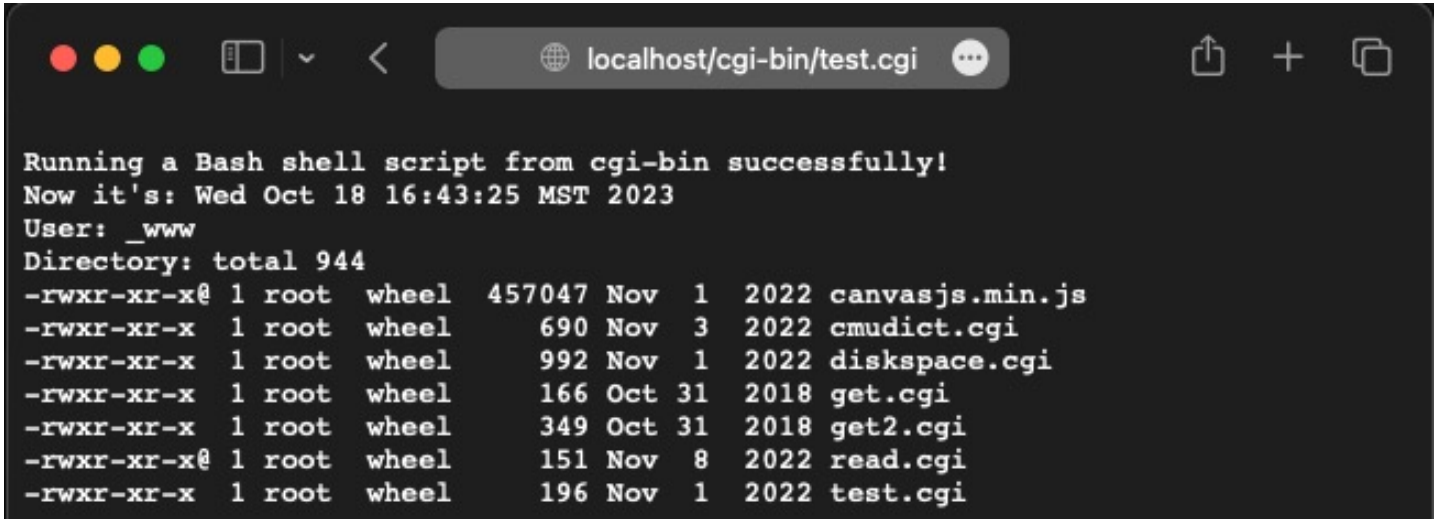
- possible: *404 Not Found: some_program.cgi doesn't exist!*

`cgi-bin` on macOS

- By default, the cgi module is turned off
- Instructions:
  1. in file `/etc/apache2/httpd.conf`
     - `sudo nano /etc/apache2/httpd.conf`
     - and uncomment line shown ☞
  2. restart apache2:
     - `sudo apachectl –k restart`

```
167LoadModule autoindex_module libexec/apache2/mod_autoindex.so
168#LoadModule asis_module libexec/apache2/mod_asis.so
169#LoadModule info_module libexec/apache2/mod_info.so
170<IfModule !mpm_prefork_module>
171        #LoadModule cgid_module libexec/apache2/mod_cgid.so
172</IfModule>
173<IfModule mpm_prefork_module>
174        #LoadModule cgi_module libexec/apache2/mod_cgi.so
175</IfModule>
176#LoadModule dav_fs_module libexec/apache2/mod_dav_fs.so
177#LoadModule dav_lock_module libexec/apache2/mod_dav_lock.so
```

`cgi-bin` on macOS



`cgi-bin` on macOS

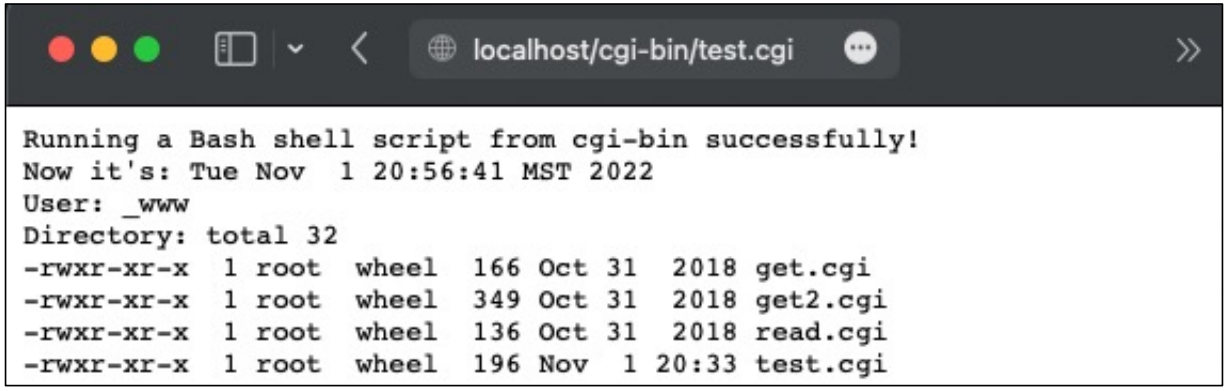
Boilerplate browser expects:

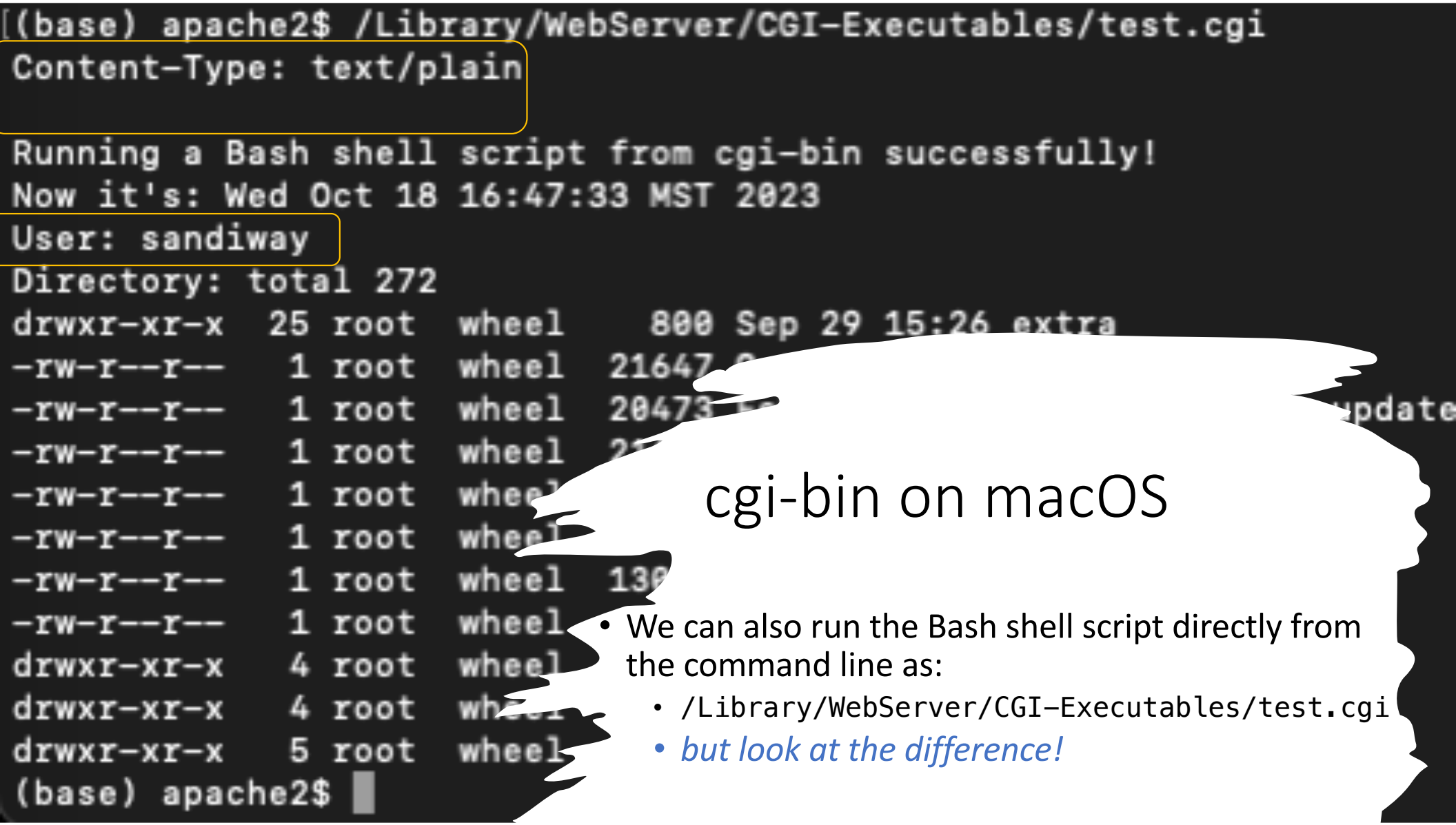
`Content-Type: text/plain`

`<blank line>`

Content

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
echo "Running a Bash shell script from cgi-bin successfully!"
echo -n "Now it's: "
date
echo -n "User: "
whoami
echo -n "Directory: "
ls -l
exit 0
```





cgi-bin on macOS

- We can also run the Bash shell script directly from the command line as:
  - `/Library/WebServer/CGI-Executables/test.cgi`
  - *but look at the difference!*

`cgi-bin` on on Ubuntu

- http://localhost/cgi-bin/
- CGI binaries directory: **`/usr/lib/cgi-bin/`**
  - `test.cgi` *must be made executable!*

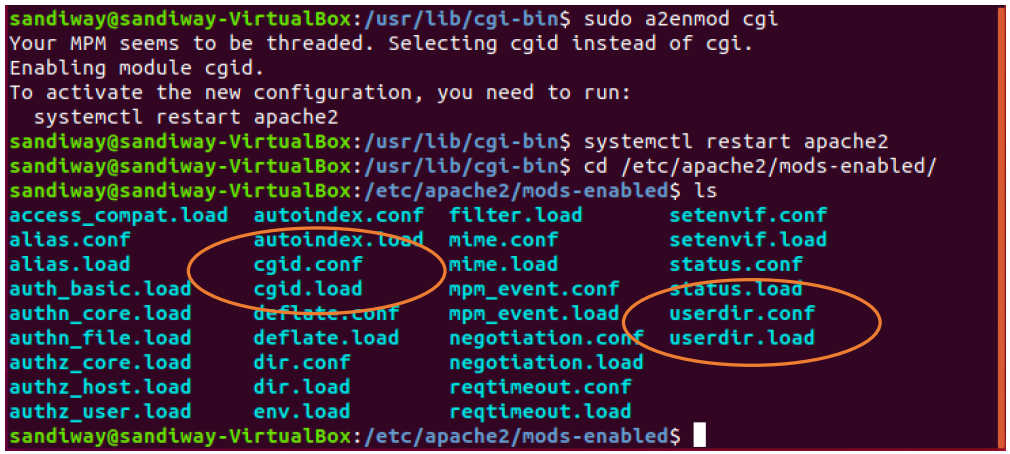
```
sandiway@sandiway-VirtualBox:/usr/lib$ ls cgi-bin
sandiway@sandiway-VirtualBox:/usr/lib$ cd cgi-bin
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ ls -l
total 0
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ sudo nano test.cgi
[sudo] password for sandiway:
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ more test.cgi
#!/bin/bash
echo "Content-Type: text/plain"
echo
echo "Run Bash script from /usr/lib/cgi-bin successfully!"
echo -n "Now: "
date
echo -n "User: "
whoami
exit 0

sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ ./test.cgi
bash: ./test.cgi: Permission denied
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ chmod 755 test.cgi
chmod: changing permissions of 'test.cgi': Operation not permitted
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ sudo chmod 755 test.cgi
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ ./test.cgi
Content-Type: text/plain

Run Bash script from /usr/lib/cgi-bin successfully!
Now: Tue Oct 30 11:24:21 MST 2018
User: sandiway
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$
```

create same test script

executable permissions needed

`cgi-bin` on Ubuntu

- Enabling `cgi-bin` on VirtualBox:
  - **`sudo a2enmod cgi`** (*enables cgid instead of cgi*)
  - directory **`/etc/apache2/mods-enabled/`**



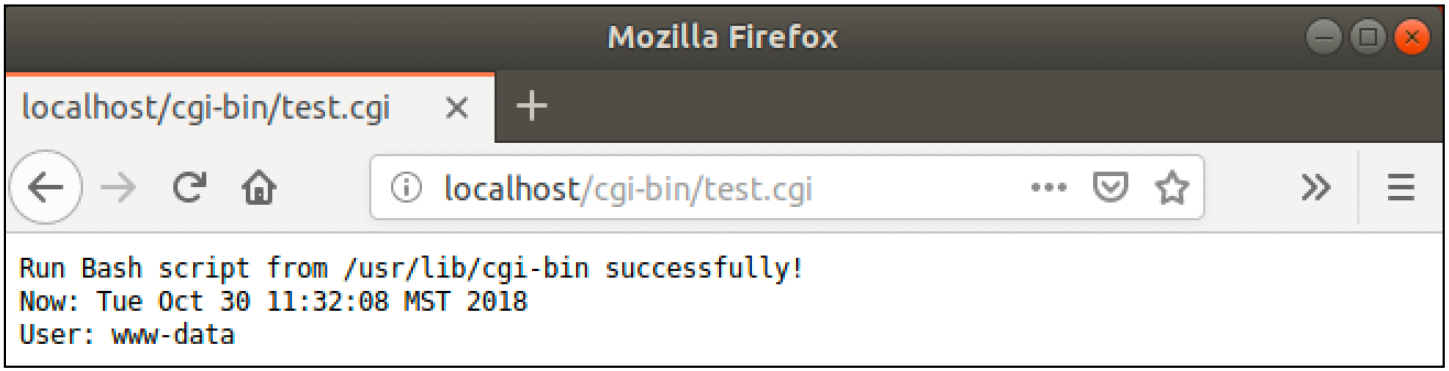
in WSL, systemctl doesn't seem to work, use
`sudo service apache2 restart`

Apache Webserver on Ubuntu

- Compare running `test.cgi` directly:

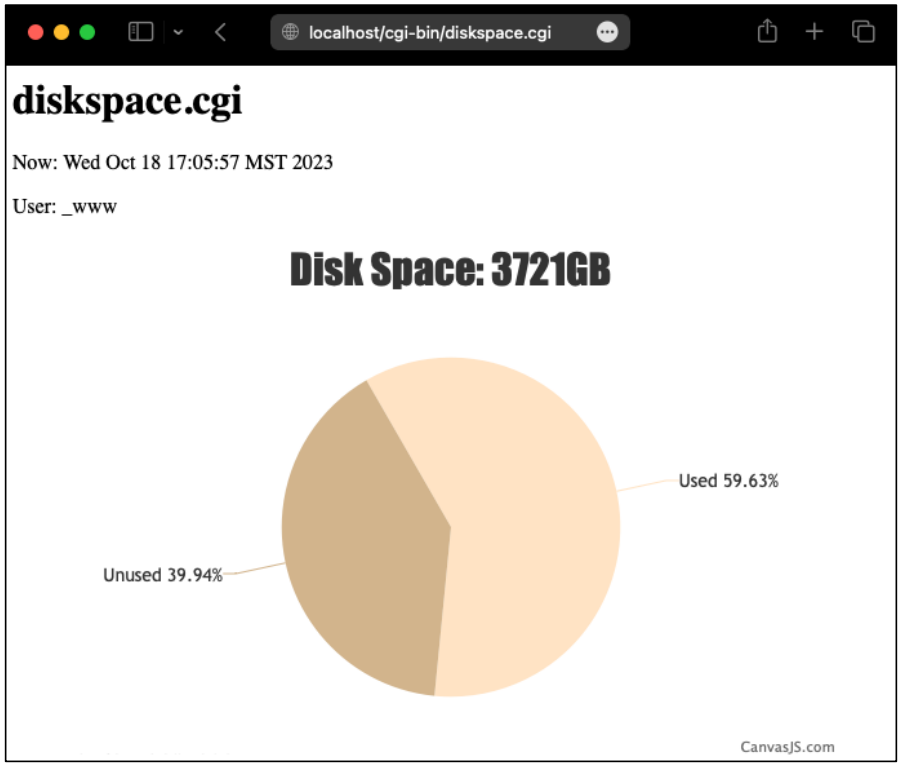
```
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ ./test.cgi
Content-Type: text/plain

Run Bash script from /usr/lib/cgi-bin successfully!
Now: Tue Oct 30 11:24:21 MST 2018
User: sandiway
```

- `http//localhost/cgi-bin/test.cgi:`



User: **www-data**
On macOS: **_www**

An Example using HTML/Javascript



- Course website:
  - `diskspace.cgi`
  - `canvasjs.min.js`
- Instead of

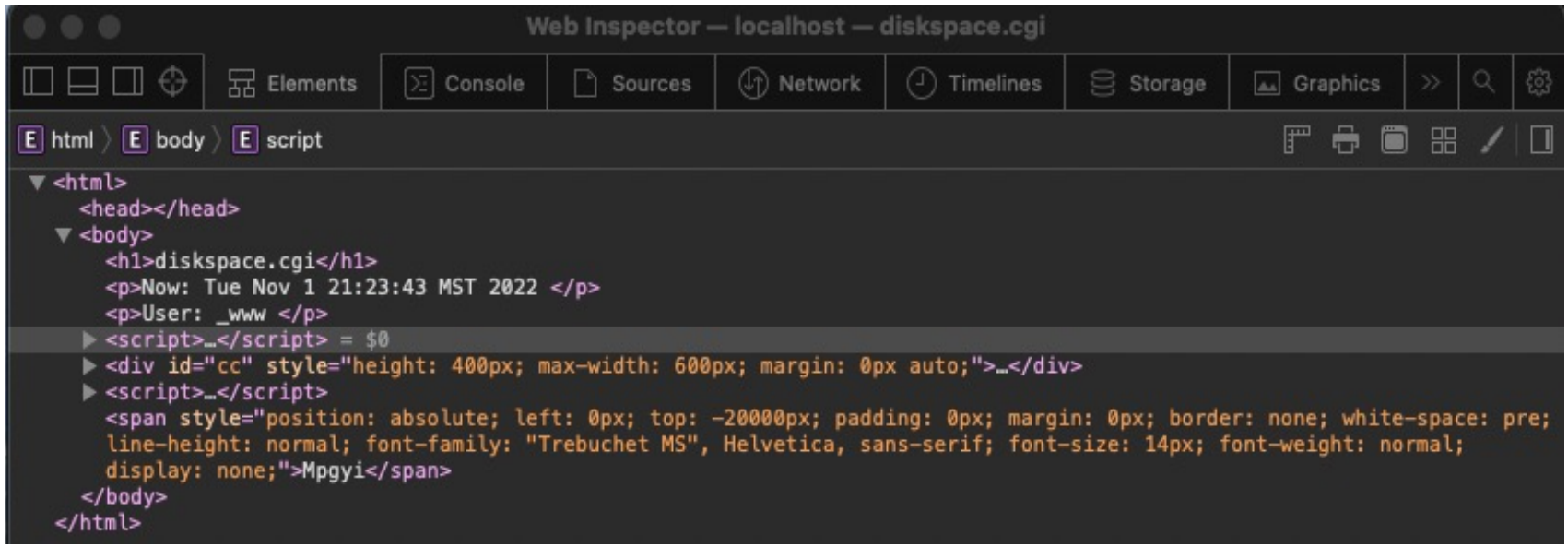
```
Content-Type: text/plain
(blank line)
```

- we use:

```
Content-Type: text/html; charset=utf-8
(blank line)
```

An Example using HTML/Javascript

- Let's look inside the source code!



An Example using HTML/Javascript

diskspace.cgi:

```
#!/bin/bash
echo "Content-Type: text/html; charset=utf-8"
echo
echo "<html><head></head>"
echo "<body><h1>diskspace.cgi</h1>"
echo -n "<p>Now: "
date
echo "</p>"
echo -n "<p>User: "
whoami
echo "</p>"
capacity=$(df -g | awk 'NR==10 {print $2}')
used=$(df -g | awk 'NR==10 {print $3}')
unused=$(df -g | awk 'NR==10 {print $4}')
```

An Example using HTML/Javascript

```
echo "<script>"
cat canvasjs.min.js
echo "</script>"
echo "<div id=\"cc\" style=\"height: 400px; max-width: 600px; margin: 0px auto;\"></div>"
echo "<script> window.onload = function() {"
echo "var chart = new CanvasJS.Chart(\"cc\", {"
echo "animationEnabled: true, title: { text: \"Disk Space: ${capacity}GB\" },"
echo "data: [{ type: \"pie\", startAngle: 240,"
echo "yValueFormatString: \"##0.00'%'\", indexLabel: \"{label} {y}\","
echo "dataPoints: ["
echo "{y: $used/$capacity*100, label: \"Used\", color: \"Bisque\"},"
echo "{y: $unused/$capacity*100, label: \"Unused\", color: \"Tan\"}"
echo "]}]}"
echo "); chart.render(); } </script></body></html>"
exit 0
```

An Example using HTML/Javascript

- `$ df –g` (*on my M1 Apple laptop*)

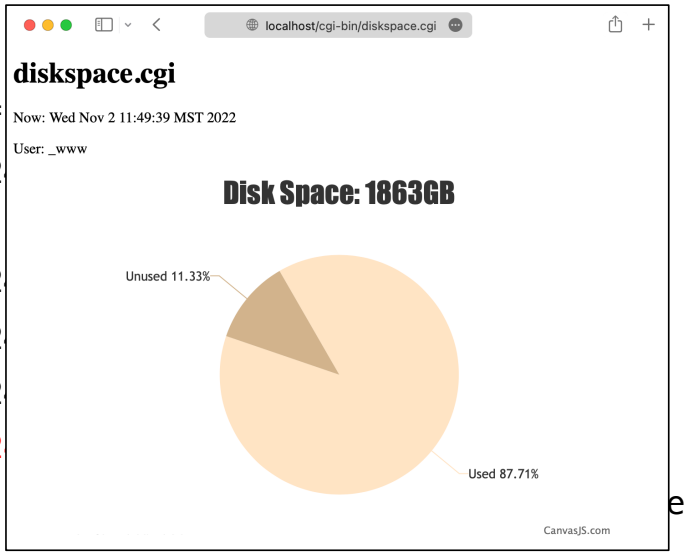
```
1. Filesystem                1G-blocks Used Available Capacity  iused       ifree %iused  Mounted on
2. /dev/disk3s1s1                 3721    9      1486     1%   387452  4292442203    0%   /
3. devfs                             0    0         0   100%      744           0  100%   /dev
4. /dev/disk3s6                   3721    0      1486     1%        0 15591697480    0%   /System/Volumes/VM
5. /dev/disk3s2                   3721    5      1486     1%     1040 15591697480    0%   /System/Volumes/Preboot
6. /dev/disk3s4                   3721    0      1486     1%       45 15591697480    0%   /System/Volumes/Update
7. /dev/disk1s2                      0    0         0     2%        1     4916120    0%   /System/Volumes/xarts
8. /dev/disk1s1                      0    0         0     2%       28     4916120    0%   /System/Volumes/iSCPreboot
9. /dev/disk1s3                      0    0         0     1%       52     4916120    0%   /System/Volumes/Hardware
10./dev/disk3s5                   3721 2219      1486    60%  4025155 15591697480    0%   /System/Volumes/Data
11.map auto_home                     0    0         0   100%        0           0     -   /System/Volumes/Data/home
```

```
• capacity=$(df –g | awk 'NR==10 {print $2}')
• used=$(df –g | awk 'NR==10 {print $3}')
• unused=$(df –g | awk 'NR==10 {print $4}')
```

An Example using HTML/Javascript

- `$ df –g` (*on my Intel Apple laptop*)

```
1.Filesystem     1G-blocks Used Available Capacity iused      if
2./dev/disk1s5s1      1863   14       211     7%  502068 2217512
3.devfs                  0    0         0   100%     666
4./dev/disk1s4        1863    1       211     1%       1 2217512
5./dev/disk1s2        1863    0       211     1%    4571 2217512
6./dev/disk1s6        1863    0       211     1%      18 2217512
7./dev/disk1s1        1863 1634       211    89% 3813194 2217512
8.map auto_home          0    0         0   100%       0
```




```
capacity=$(df -g | awk 'NR==7 {print $2}')
used=$(df -g | awk 'NR==7 {print $3}')
unused=$(df -g | awk 'NR==7 {print $4}')
```

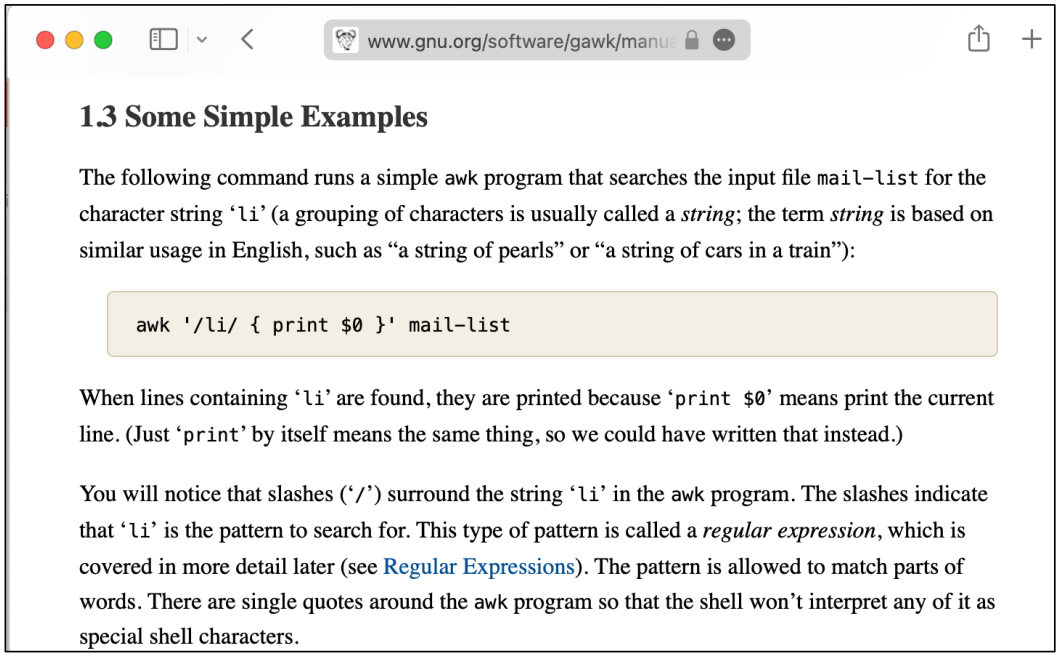
What is awk?

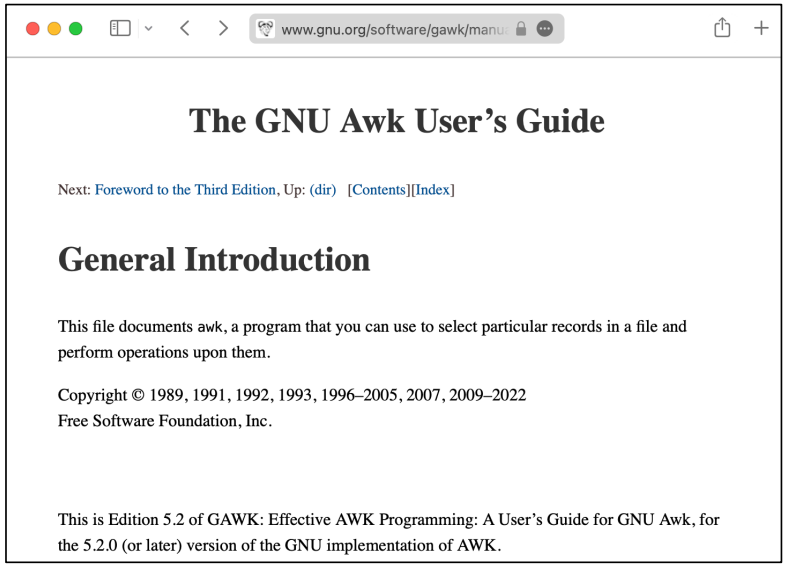
- record:
  - basically, a line
- NR:
  - NR is the number of records read so far
- $1:
  - You use a dollar sign (‘$’) to refer to a field in an awk program, followed by the number of the field you want.
  - Thus, $1 refers to the first field, $2 to the second, and so on.

- Idea:
  - When awk reads an input record, the record is automatically *parsed* or separated by the awk utility into chunks called *fields*.
  - By default, fields are separated by *whitespace*, like words in a line.

What is awk?

https://www.gnu.org/software/gawk/manual/gawk.html





Documentation

http://httpd.apache.org/docs/current/