

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.

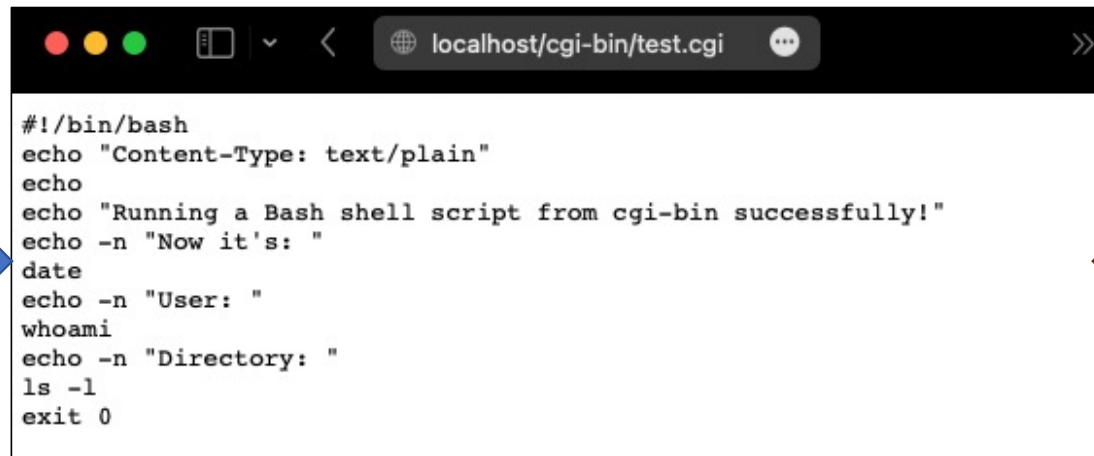
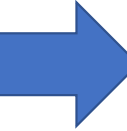
The whoami utility displays your effective user ID as a name.

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
```

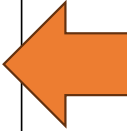
- Put it in directory
 - /Library/WebServer/CGI-Executables
 - sudo needed (owned by root)
- URL:
 - <http://localhost/cgi-bin/test.cgi>

test.cgi must be executable and readable
chmod 755



```
#!/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
```

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



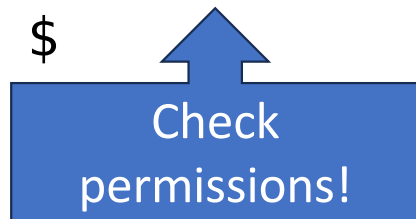
cgi-bin on macOS

```
$ 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!*

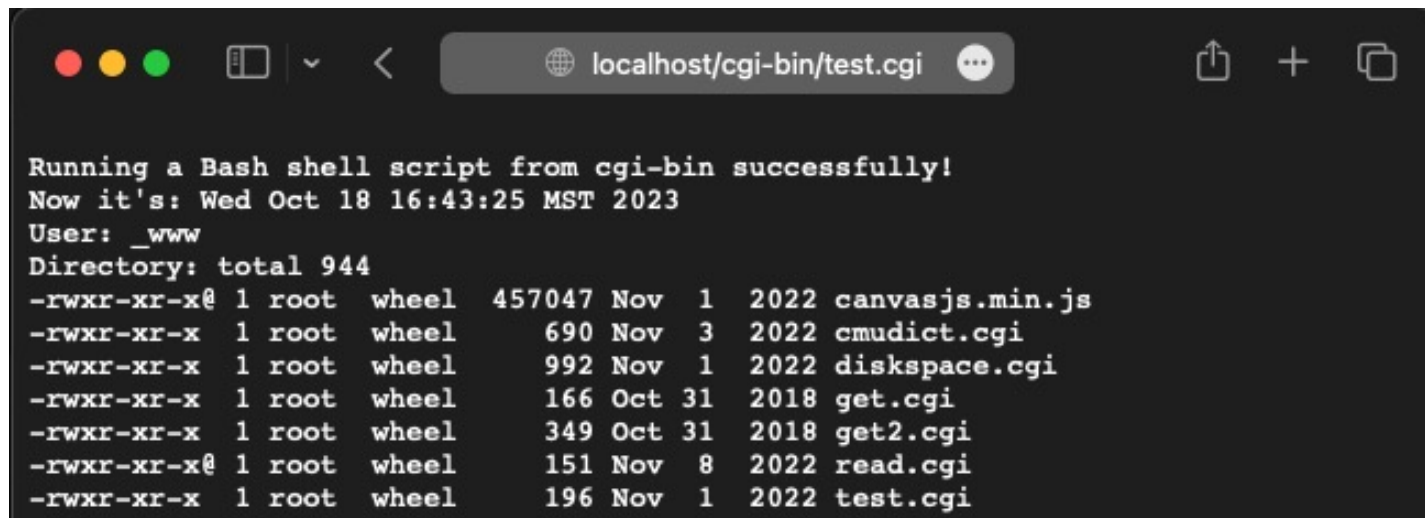


cgi-bin on macOS

- 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



The image shows a screenshot of a web browser window with a dark theme. The address bar at the top displays "localhost/cgi-bin/test.cgi". Below the address bar, the terminal output of a CGI script is shown. The output includes a success message, the current date and time, the user name, and a directory listing of files in the cgi-bin directory.

```
Running a Bash shell script from cgi-bin successfully!  
Now it's: Wed Oct 18 16:43:25 MST 2023  
User: _www  
Directory: total 944  
-rwxr-xr-x@ 1 root  wheel  457047 Nov  1  2022 canvasjs.min.js  
-rwxr-xr-x  1 root  wheel    690 Nov  3  2022 cmudict.cgi  
-rwxr-xr-x  1 root  wheel    992 Nov  1  2022 diskspace.cgi  
-rwxr-xr-x  1 root  wheel    166 Oct 31  2018 get.cgi  
-rwxr-xr-x  1 root  wheel    349 Oct 31  2018 get2.cgi  
-rwxr-xr-x@ 1 root  wheel    151 Nov  8  2022 read.cgi  
-rwxr-xr-x  1 root  wheel    196 Nov  1  2022 test.cgi
```

cgi-bin on macOS

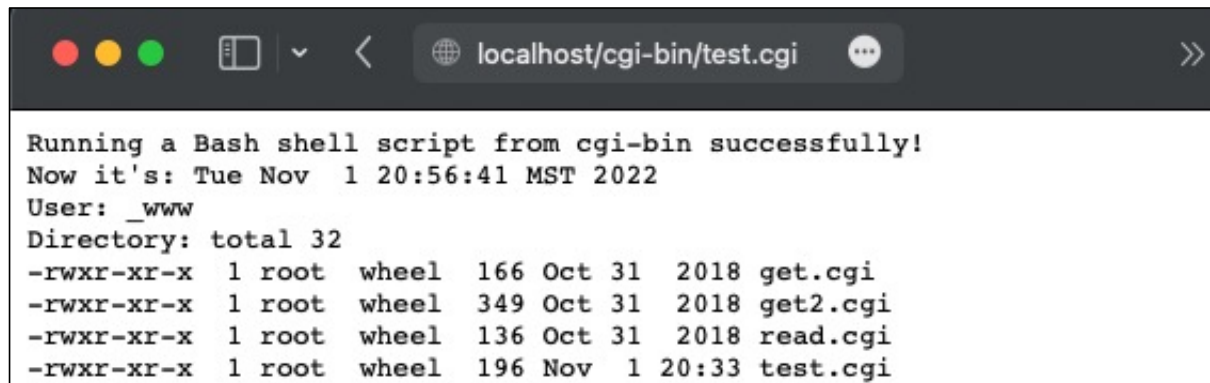
Boilerplate browser expects:

Content-Type: text/plain

<blank line>

Content

```
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
```



The screenshot shows a web browser window with the address bar displaying 'localhost/cgi-bin/test.cgi'. The main content area shows the output of the script, which includes a success message, the current date and time, the user name, and a directory listing of the current directory.

```
Running a Bash shell script from cgi-bin successfully!
Now it's: Tue Nov  1 20:56:41 MST 2022
User: _www
Directory: total 32
-rwxr-xr-x  1 root  wheel  166 Oct 31  2018 get.cgi
-rwxr-xr-x  1 root  wheel  349 Oct 31  2018 get2.cgi
-rwxr-xr-x  1 root  wheel  136 Oct 31  2018 read.cgi
-rwxr-xr-x  1 root  wheel  196 Nov  1 20:33 test.cgi
```

```
(base) apache2$ /Library/WebServer/CGI-Executables/test.cgi
```

```
Content-Type: text/plain
```

```
Running a Bash shell script from cgi-bin successfully!
```

```
Now it's: Wed Oct 18 16:47:33 MST 2023
```

```
User: sandiway
```

```
Directory: total 272
```

```
drwxr-xr-x  25 root  wheel   800 Sep 29 15:26 extra
```

```
-rw-r--r--   1 root  wheel 21647 0
```

```
-rw-r--r--   1 root  wheel 20473 1 update
```

```
-rw-r--r--   1 root  wheel 21
```

```
-rw-r--r--   1 root  wheel
```

```
-rw-r--r--   1 root  wheel
```

```
-rw-r--r--   1 root  wheel 130
```

```
-rw-r--r--   1 root  wheel
```

```
drwxr-xr-x   4 root  wheel
```

```
drwxr-xr-x   4 root  wheel
```

```
drwxr-xr-x   5 root  wheel
```

```
(base) apache2$
```

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/**

```
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ sudo a2enmod cgi
Your MPM seems to be threaded. Selecting cgid instead of cgi.
Enabling module cgid.
To activate the new configuration, you need to run:
  systemctl restart apache2
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ systemctl restart apache2
sandiway@sandiway-VirtualBox:/usr/lib/cgi-bin$ cd /etc/apache2/mods-enabled/
sandiway@sandiway-VirtualBox:/etc/apache2/mods-enabled$ ls
access_compat.load  autoindex.conf  filter.load      setenvif.conf
alias.conf          autoindex.load  mime.conf        setenvif.load
alias.load          cgid.conf       mime.load        status.conf
auth_basic.load    cgid.load       mpm_event.conf  status.load
auth_core.load     deflate.conf    mpm_event.load  userdir.conf
auth_file.load     deflate.load    negotiation.conf userdir.load
auth_core.load     dir.conf        negotiation.load
auth_host.load     dir.load        reqtimeout.conf
auth_user.load     env.load        reqtimeout.load
sandiway@sandiway-VirtualBox:/etc/apache2/mods-enabled$
```

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

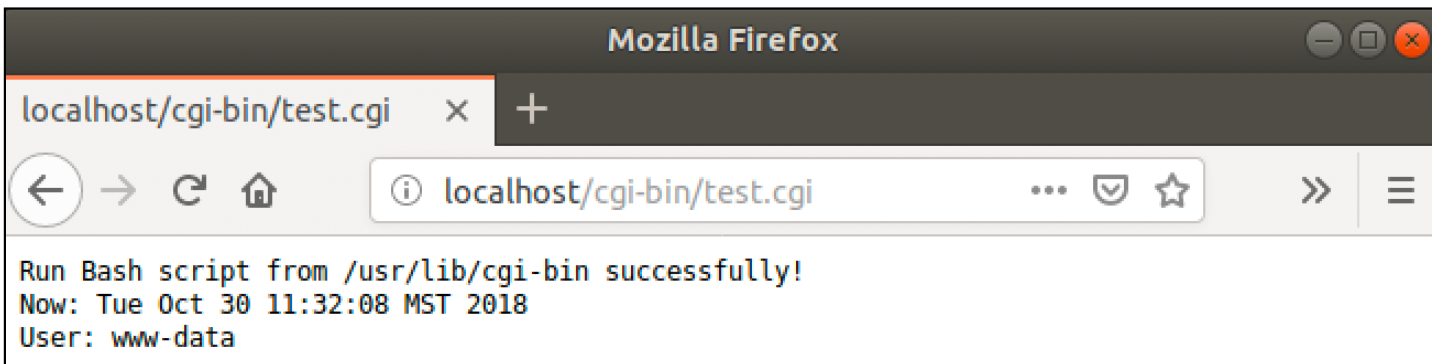
Apache Webserver on Ubuntu

- Compare running `test.cgi` directly:

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

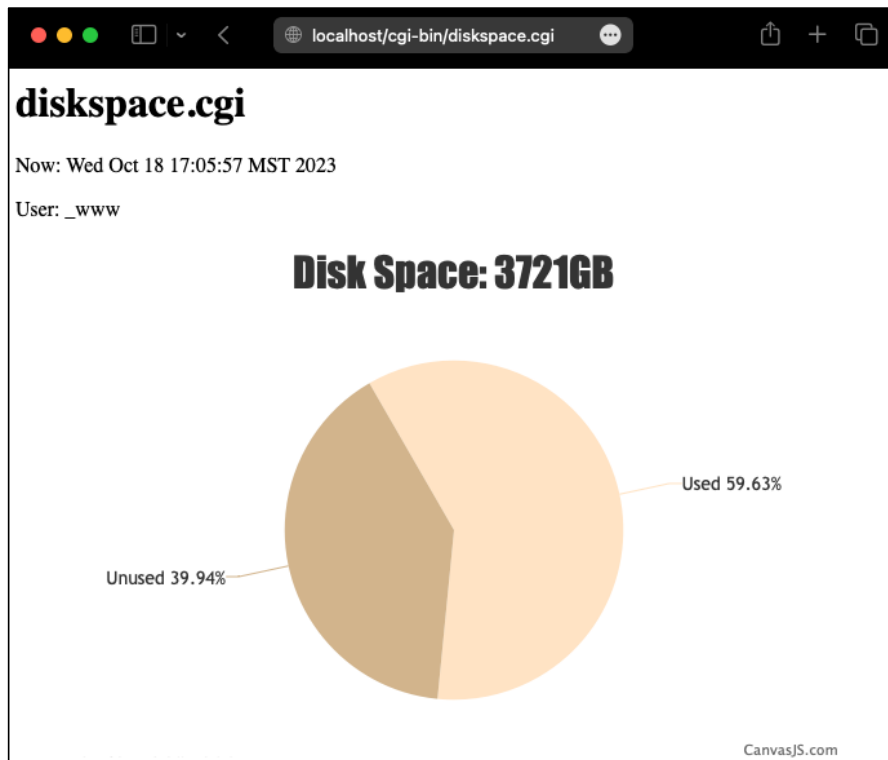
Run Bash script from /usr/lib/cgi-bin successfully!
Now: Tue Oct 30 11:24:21 MST 2018
User: sandiway
```

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



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

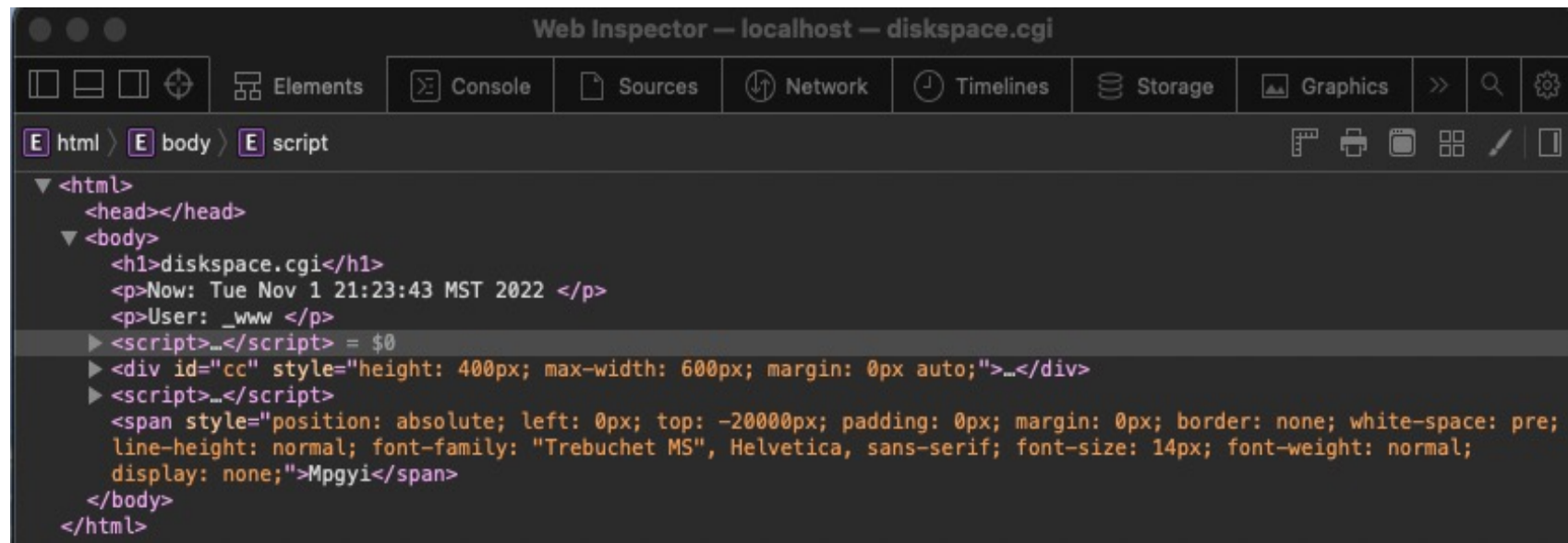
An Example using HTML/Javascript



- 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!



```
Web Inspector — localhost — diskspace.cgi
Elements Console Sources Network Timelines Storage Graphics
html > body > script
<html>
  <head></head>
  <body>
    <h1>diskspace.cgi</h1>
    <p>Now: Tue Nov 1 21:23:43 MST 2022 </p>
    <p>User: _www </p>
    <script>_</script> = $0
    <div id="cc" style="height: 400px; max-width: 600px; margin: 0px auto;">_</div>
    <script>_</script>
    <span style="position: absolute; left: 0px; top: -20000px; padding: 0px; margin: 0px; border: none; white-space: pre; line-height: normal; font-family: "Trebuchet MS", Helvetica, sans-serif; font-size: 14px; font-weight: normal; display: none;">Mpgyi</span>
  </body>
</html>
```

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)

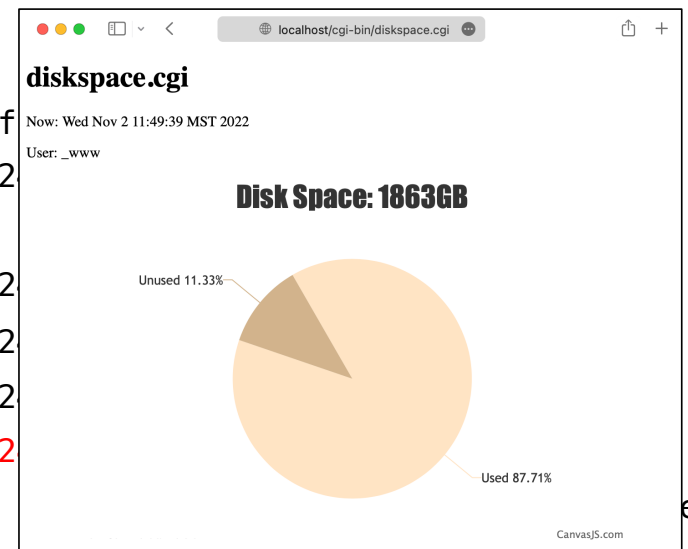
1. Filesystem	1G-blocks	Used	Available	Capacity	iused	ifree	%iused	Mounted on
2. /dev/disk3s1s1	3721	9	1486	1%	387452	4292442203	0%	/
3. devfs	0	0	0	100%	744	0	100%	/dev
4. /dev/disk3s6	3721	0	1486	1%	0	15591697480	0%	/System/Volumes/VM
5. /dev/disk3s2	3721	5	1486	1%	1040	15591697480	0%	/System/Volumes/Preboot
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	used	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	



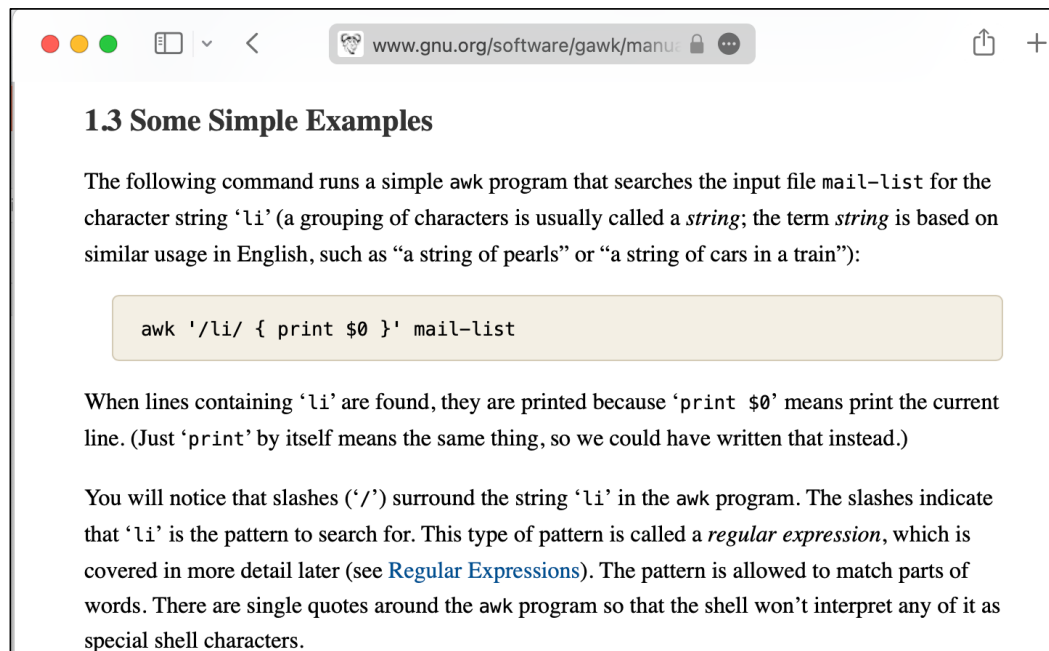
```
capacity=$(df -g | awk 'NR==7 {print $2}')  
used=$(df -g | awk 'NR==7 {print $3}')  
unused=$(df -g | awk 'NR==7 {print $4}')
```

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>



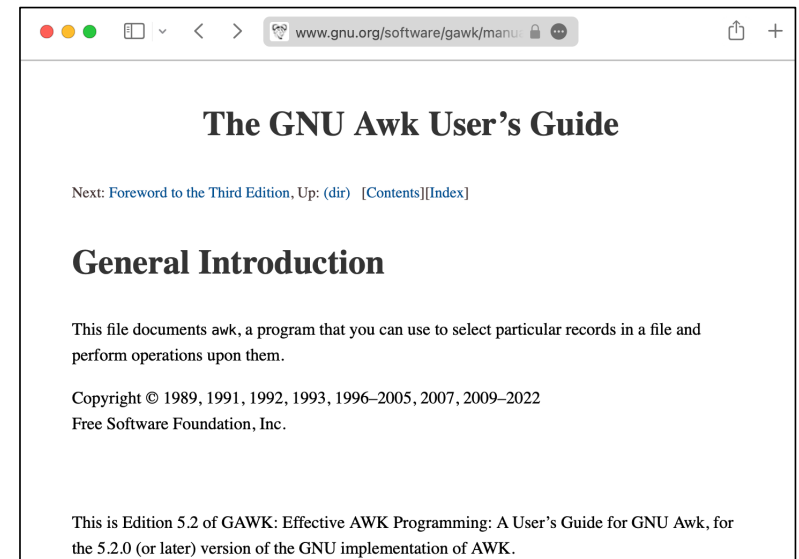
1.3 Some Simple Examples

The following command runs a simple awk program that searches the input file `mail-list` for the character string `'li'` (a grouping of characters is usually called a *string*; the term *string* is based on similar usage in English, such as “a string of pearls” or “a string of cars in a train”):

```
awk '/li/ { print $0 }' mail-list
```

When lines containing `'li'` are found, they are printed because `'print $0'` means print the current line. (Just `'print'` by itself means the same thing, so we could have written that instead.)

You will notice that slashes (`'/'`) surround the string `'li'` in the awk program. The slashes indicate that `'li'` is the pattern to search for. This type of pattern is called a *regular expression*, which is covered in more detail later (see [Regular Expressions](#)). The pattern is allowed to match parts of words. There are single quotes around the awk program so that the shell won't interpret any of it as special shell characters.



The GNU Awk User's Guide

Next: [Foreword to the Third Edition](#), Up: [\(dir\)](#) [[Contents](#)][[Index](#)]

General Introduction

This file documents awk, a program that you can use to select particular records in a file and perform operations upon them.

Copyright © 1989, 1991, 1992, 1993, 1996–2005, 2007, 2009–2022
Free Software Foundation, Inc.

This is Edition 5.2 of GAWK: Effective AWK Programming: A User's Guide for GNU Awk, for the 5.2.0 (or later) version of the GNU implementation of AWK.

Documentation

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

The screenshot shows the Apache HTTP Server Version 2.4 Documentation website. The browser address bar displays the URL <http://httpd.apache.org/docs/current/>. The page features the Apache logo and the text "APACHE HTTP SERVER PROJECT" and "Apache HTTP Server Version 2.4". Navigation links include "Modules", "Directives", "FAQ", "Glossary", and "Sitemap". The breadcrumb trail is "Apache > HTTP Server > Documentation". The main heading is "Apache HTTP Server Version 2.4 Documentation". Below the heading, there are language options: "Available Languages: da | de | en | es | fr | ja | ko | pt-br | ru | tr | zh-cn". A search box with a "Google Search" button is present. The page is organized into three columns of links:

- Release Notes**
 - [New features with Apache 2.3/2.4](#)
 - [New features with Apache 2.1/2.2](#)
 - [New features with Apache 2.0](#)
 - [Upgrading to 2.4 from 2.2](#)
 - [Apache License](#)
- Reference Manual**
 - [Compiling and Installing](#)
 - [Starting](#)
 - [Stopping or Restarting](#)
 - [Run-time Configuration Directives](#)
 - [Modules](#)
 - [Multi-Processing Modules \(MPMs\)](#)
 - [Filters](#)
 - [Handlers](#)
- Users' Guide**
 - [Getting Started](#)
 - [Binding to Addresses and Ports](#)
 - [Configuration Files](#)
 - [Configuration Sections](#)
 - [Content Caching](#)
 - [Content Negotiation](#)
 - [Dynamic Shared Objects \(DSO\)](#)
 - [Environment Variables](#)
 - [Log Files](#)
 - [Mapping URLs to the Filesystem](#)
 - [Performance Tuning](#)
 - [Security Tips](#)
 - [Server-Wide Configuration](#)
 - [SSL/TLS Encryption](#)
 - [Suexec Execution for CGI](#)
- How-To / Tutorials**
 - [Authentication and Authorization](#)
 - [Access Control](#)
 - [CGI: Dynamic Content](#)
 - [.htaccess files](#)
 - [Server Side Includes \(SSI\)](#)
 - [Per-user Web Directories \(public_html\)](#)
 - [Reverse proxy setup guide](#)
 - [HTTP/2 guide](#)
- Platform Specific Notes**
 - [Microsoft Windows](#)
 - [RPM-based Systems \(Redhat / CentOS / Fedora\)](#)
 - [Novell NetWare](#)
 - [EBCDIC Port](#)
- Other Topics**