COMP 307

Principles of Web Development

Lecture 12

Unit 4 – Servers

About Webservers and the SOCS web server

Contents

Class Outline

- About webservers
- The SOCS webserver
- CGI & HTML Forms
- Server-side execution examples
  - The C Language
  - Python
  - Bash

Readings

- Internet and World Wide Web Textbook
  - Chapter 21
- The Full Stack Developer
  - Chapter 10
- Internet Resources
  - https://www.tutorialspoint.com/python/python_cgi_programming.htm
  - https://jkorpela.fi/forms/cgic.html

About Webservers

About webservers and the SOCS webserver

Contents

An Internet Server

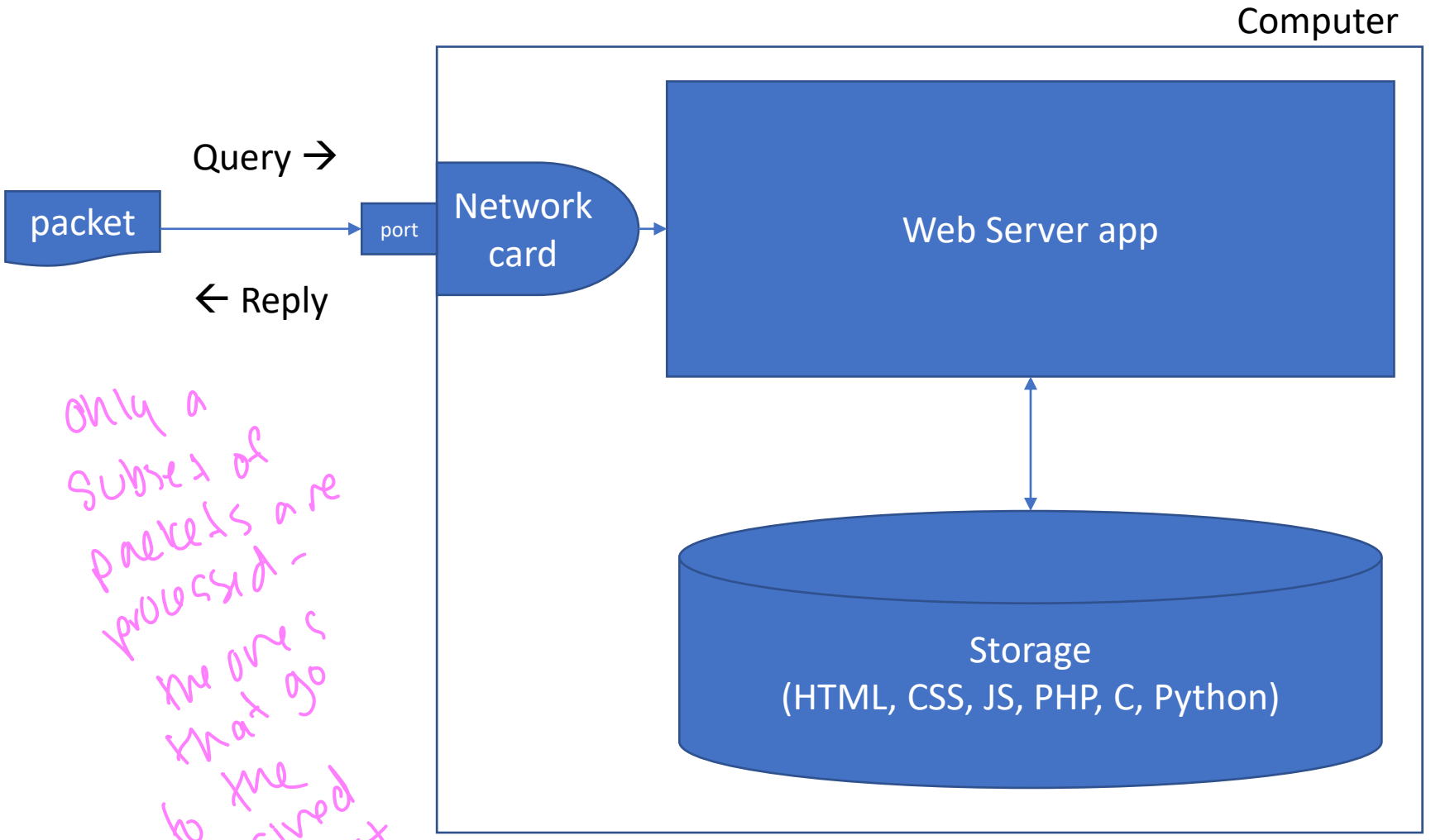
- A program that runs on a computer
- It listens to the port that receives packets from the network card (a device installed in the computer to connect to the internet, like the Wi-Fi your laptop uses)
- Uses REST in its communication protocol
  - **Re**presentational **S**tate **T**ransfer
  - Transmitting state & query information
  - HTTP & HTTPS is the protocol
- It reads the packet's query string and:
  - Returns a reply to the sender, or
  - Returns an ACK packet (acknowledgement message) to the sender

Web servers job is to implement rest + http

COMP 307
Principles of Web Development

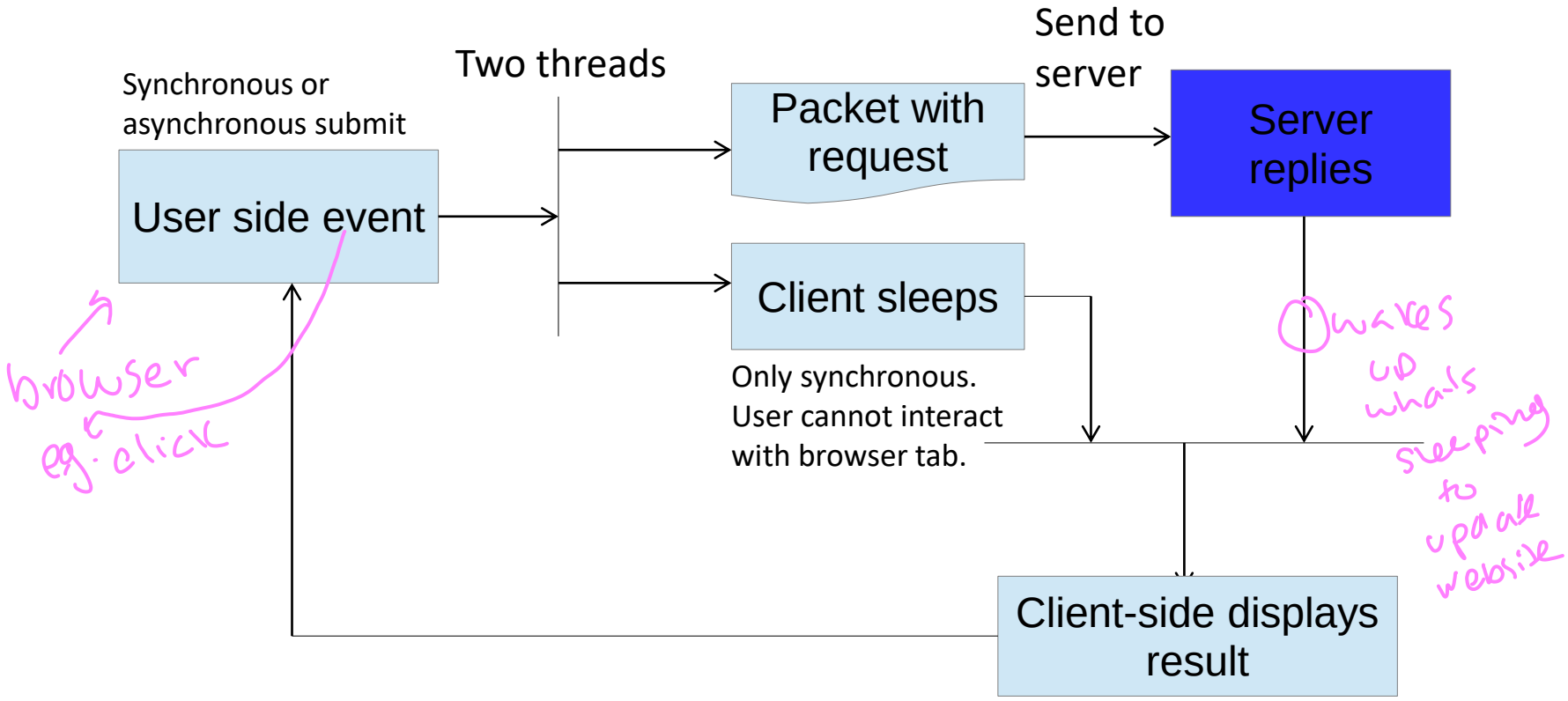
To browser for rendering

An Internet Server



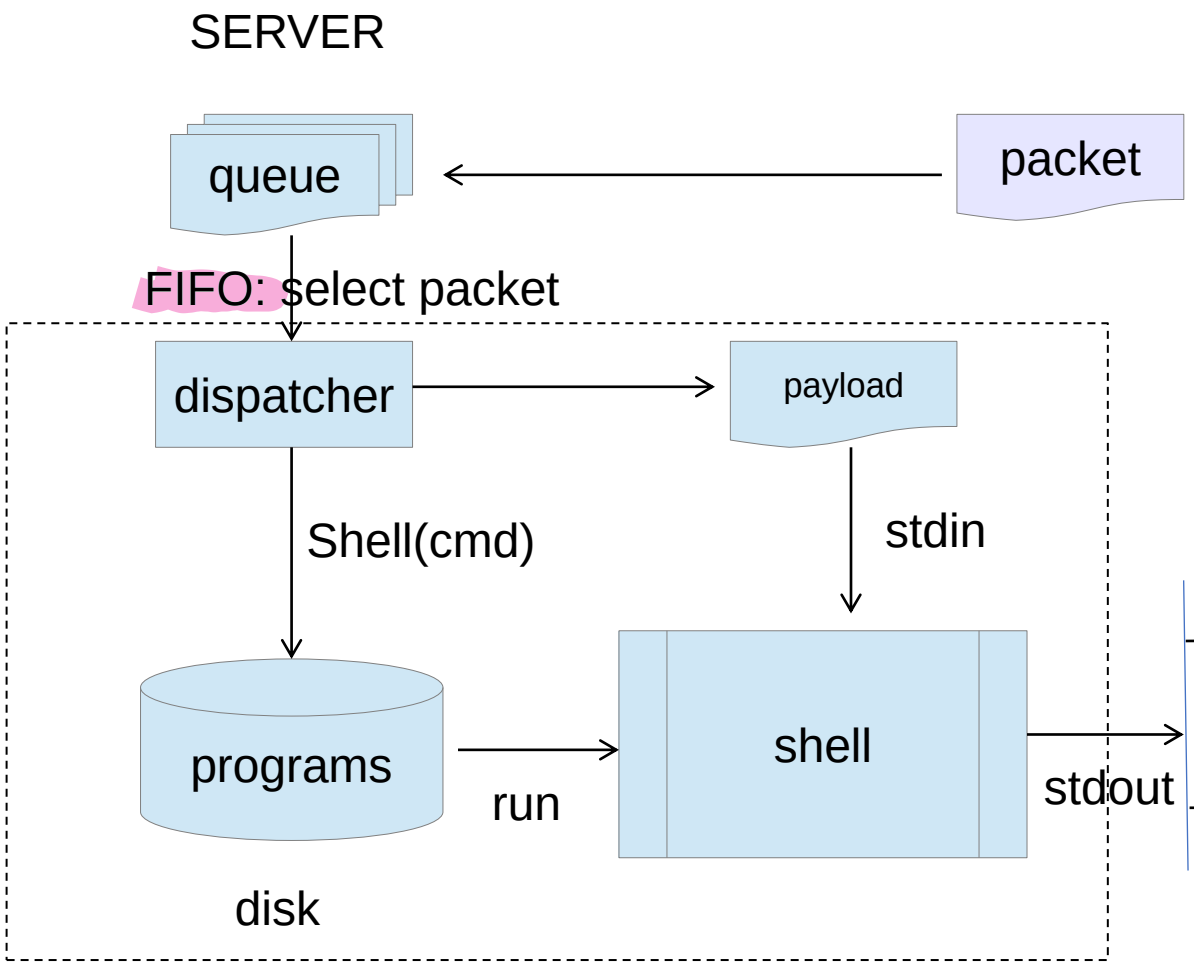
Only a subset of packets are processed - the ones that go to the desired port

Client-side REST



"client" means the process that made the request. This could be the browser tab itself or a function.

Server-side REST



List of well know port numbers

Port	Description	Port	Description
1	TCP Port Service Multiplexer (TCPMUX)	18	Message Send Protocol (MSP)
7	ECHO	21	FTP -- Control
20	FTP – Data	23	Telnet
22	SSH Remote Login Protocol	29	MSG ICP
25	Simple Mail Transfer Protocol (SMTP)	42	Host Name Server (Nameserv)
37	Time	49	Login Host Protocol (Login)
43	WhoIs	69	Trivial File Transfer Protocol (TFTP)
53	Domain Name System (DNS)	79	Finger
70	Gopher Services	103	X.400 Standard
80	HTTP	109	POP2
108	SNA Gateway Access Server	115	Simple File Transfer Protocol (SFTP)
110	POP3	119	Newsgroup (NNTP)
118	SQL Services	139	NetBIOS Datagram Service
137	NetBIOS Name Service	150	NetBIOS Session Service
143	Interim Mail Access Protocol (IMAP)	161	SNMP
156	SQL Server	190	Gateway Access Control Protocol
179	Border Gateway Protocol (BGP)	197	Directory Location Service (DLS)
194	Internet Relay Chat (IRC)		
389	Lightweight Directory Access Protocol (LDAP)		
396	Novell Netware over IP	443	HTTPS
444	Simple Network Paging Protocol (SNPP)	445	Microsoft-DS
458	Apple QuickTime	546	DHCP Client
547	DHCP Server	563	SNEWS
569	MSN	1080	Socks

Important Trends

- Apache-based solutions
  - It started with this one
  - 90% of websites use this
  - Best plug-and-play environment
- NodeJS-based solutions
  - Trending as an alternate to Apache for dynamic elements
  - Less plug-and-play friendly: focus on JavaScript
- Django-based solutions
  - Popular with Python programmers
- Microsoft-based solutions
  - Popular with Microsoft devotees

still plug + playable

What is a payload?

- It is that part of the packet that contains the information (message, query, html).
- Packet = control_info + payload + error_correction_info
  - Control_info = from, to, type, etc.
  - Eroor_correction_info = used to recover data transition mistakes

HTTP Query Packet

(from client to server)

path to program

program we want

GET /index.html HTTP/1.1 ← Request line

Date: Thu, 20 May 2014 21:12:15 GMT
Connection: close ← General headers

Host: www.someplace.com
From: bob@someotherplace.som
Accept: text/html, text/plain
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) ← Request headers

Content-Language: en
Content-Length:22 ← Entity headers

The empty line

name=john+smith&age=10 ← Message body

- CGI string
- XML
- JSON
- custom

arguments being sent

All the above fields can be extracted by the server

client → server

HTTP Response Packet

(from server to client)

end-end ack

```
HTTP/1.1  200  OK
Date: Thu, 20 May 2014 21:12:15 GMT
Connection: close
Server: Apache/2.3.7
Accept-Ranges: bytes
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Type: text/html
Content-Length:170
Last-Modified: Tue, 18 May 2014 10:14:49 GMT

<html>
        <head>
                <title> ... </title>
        </head>
        <body>
                <p>Your page</p>
        </body>
</html>
```

Status line

General headers

Request headers

Entity headers

The empty line

Message body

HTML, XML, JSON, Plain text, binary

Server → client

All the above fields can be extracted by the client

Web Development Stack

- Client-side browser app
  - HTML, CSS, JS (React, TypeScript, Vue.js, JQuery), etc.
- Communication Method
  - CGI, AJAX, etc.
- Webserver
  - Apache, Node.js, etc.
- Server-side app
  - Java, JavaScript, C, Python, PHP, Perl, etc.

SOCS Resources

About webservers and the SOCS webserver

Contents

SOCS Environment

- Apache server
- public_html
- www.cs.mcgill.ca/~yourCSusername
- cgi-bin
- chmod 755
- Built-in support for:
  - Python, C, Bash, PHP
  - Static webpages with synchronous and asynchronous CGI communication
  - CAN request for additional environments (eg Nodejs)

mkdir public_html and apache sees it

access w/ this

chmod home dir to executable only

inside public_html includes all executables

SQL

SOCS Environment

- The mimi directory public_html is like htdocs in XAMPP. It becomes the root internet directory.
- For added security, compiled programs must be stored within the directory cgi-bin (a subdirectory of public_html).
- SOCS information
  - https://www.cs.mcgill.ca/docs/tutorials/website/
  - https://www.cs.mcgill.ca/docs/tutorials/django/

SOCS Web Servers

Server:

- mimi.cs.mcgill.ca
- Lab & server share the same hard drive
  - BUT, you must compile separately
  - BUT, they have different libraries and versions installed

Therefore, pick one server and use that one all the time.

SOCS Web Directories

Important directories:

- HOME/public_html
  - Where your HTML, PHP, CSS and JS can live
  - You can make sub-directories to organize
- HOME/public_html/cgi-bin
  - Where your compiled programs and scripts live
    - a.out (named a.cgi) or program.py (can keep .py extension)

SOCS Web Permissions

Your directories need to be accessible to the public

- Minimum permission: chmod +x
- Also good to do: chmod +r for public_html
- These directory permissions must be set:
  - HOME +x
  - public_html +x +r
  - cgi-bin +x

Permissions needed are:
~ ($HOME) : 711
~/public_html : 711
~/public_html/cgi-bin : 711
~/public_html/cgi-bin/cgi_file : 755

SOCS Web Error Messages

You can see the errors messages from your website by:

- You would need to use **VPN** and run your scripts from
  - http://mimi.cs.mcgill.ca/~USERNAME/cgi-bin
- Then run the **geterrors.cgi** from the same host The site internally is : https://cgi64-dev.cs.mcgill.ca/cgi-bin/geterrors.cgi

User's can test their code running: https://cgi64-dev.cs.mcgill.ca/~username/cgi-bin/cgi-program.cgi
If there are errors, then they can troubleshoot with the above geterrors url (please note it is **cgi64-dev** and NOT cgi-dev)

IMPORTANT

If you had logged in once to mimi using your first.last@mail.mcgill.ca....

Then, the SOCS setup may not work for you!!

You will have the McGill setup instead ☹

Email: help@cs.mcgill.ca to have your account reset.

Demo

- SSH
- SFTP
- Putty
- WinSCP
- Filezilla Client

sftp fgomppa@mimi.cs.mcgill.ca
→ login
→ lcd = local cd
get path/filename and will transfer
put path/filename send to server

SOCS and C Programming

About webservers and the SOCS webserver

Contents

Why C?

```
<form action="a.out" method="get">
```

Compiled languages are faster than interpreted languages.

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)	Database	Notes
Google[2]	1,600,000,000	JavaScript, TypeScript	C, C++, Go,[3] Java, Python	Bigtable,[4] MariaDB[5]	The most used search engine in the world
Facebook	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D,[6] XHP,[7] Haskell[8]	MariaDB, MySQL,[9] HBase, Cassandra[10]	The most visited social networking site
YouTube	1,100,000,000	JavaScript	C, C++, Python, Java,[11] Go[12]	Vitess, BigTable, MariaDB[5][13]	The most visited video sharing site
Yahoo	750,000,000	JavaScript	PHP	PostgreSQL, HBase, Cassandra, MongoDB,[14]	
Amazon	500,000,000	JavaScript	Java, C++, Perl[15]	PostgreSQL, RDS, RDS Aurora[16]	Popular Internet shopping site
Wikipedia	475,000,000	JavaScript	PHP	MariaDB[17]	"MediaWiki" is programmed in PHP; free online encyclopedia
Twitter	290,000,000	JavaScript	C++, Java[18], Scala[19], Ruby	MySQL[20]	Popular social network.
Bing	285,000,000	JavaScript	C++, C#	Microsoft SQL Server, Cosmos DB	Search engine from Microsoft.
eBay	285,000,000	JavaScript	Java,[21] JavaScript,[22] Scala[23]	Oracle Database	Online auction house.
MSN	280,000,000	JavaScript	C#	Microsoft SQL Server	An email client, for simple use. Previously known as "messenger", not to be confused with Facebook's messaging platform.
LinkedIn	260,000,000	JavaScript	Java, JavaScript,[24] Scala	Voldemort[25]	World's largest professional network.
Pinterest	250,000,000	JavaScript	Python (Django),[26] Erlang	MySQL, Redis [27]	Search engine for ideas.
WordPress.com	240,000,000	JavaScript	PHP	PostgreSQL, HBase, Cassandra, MongoDB,[14]	Website manager software.

Post vs Get Data Structures

GET

Uses shell memory for the query

(Assumes public & short strings)

POST

Uses STDIN for the query

Uses shell memory for meta data

(Assumes private & long strings)

Backend using C and GET

```
#include <stdlib.h>

char *data = getenv("QUERY_STRING");

sscanf(data,"x=%d&y=%d",&a,&b);
```

Posted in shell

shell memory variable pointer

or parse char by char if complicated.

sscanf works for %d, %c, %f input, NOT for %s (unless spaces)

strtok() can also be used.

Backend using C and POST

```
#include <stdlib.h>
char string[200];
char c;
int a = 0;
int n = atoi(getenv("CONTENT_LENGTH"));

fgets(string,n+1,stdin);
```

Input length posted

Sent to stdin!

Version 1

Backend using C and POST

Sent to stdin!

Input length posted

```
#include <stdlib.h>
char string[200];
char c;
int a = 0;
int n = atoi(getenv("CONTENT_LENGTH"));

while ((c = getchar()) != EOF && a<n+1)
{
      if (c!='+') string[a]=c;
         else string[a]=' ';
      a++;
}

string[a] = '\0';
```

replace + with space

char by char

Version 2

Backend using C to send Reply

```
#include <stdio.h>   #include <stdlib.h>
int main(void)
{
        FILE *f = fopen("data.txt","r");
        int ch;

        printf("Content-Type:text/html\n\n");
        printf("<html>");

        if (f==NULL)
        {
                printf("<head><title>ERROR</title></head>");
                printf("<body><p>Unable to open file!</p></body>");
        }
        else
        {
                while((ch=fgetc(f)) != EOF) putchar(ch);

                fclose(f);
        }

        printf("</html>");

        return 0;
}
```

Payload's extra blank line…

keyword

data type

what am i sending back

2 CR

printf's are sent back to browser

Need to output HTML

Compiling C Programs for the web

- Pick mimi and stick with it
- Use GCC compiler
- Change executable file extension to .cgi
- Copy program to cgi-bin
- Make sure chmod +x of the .cgi program
- Make sure the program is made to work with POST or GET and make sure the <form> agrees
- The action attribute's path must lead to cgi-bin

C and CGI

- vi program.c
- gcc program.c
- cp a.out ./cgi-bin/a.cgi
- chmod +x a.cgi

```
#include <stdlib.h>
#include <stdio.h>
int main() {
        char array[200];
        int n = atoi(getenv("CONTENT_LENGTH"));
        fgets(array,n,stdin);

        printf("Content-Type: text/html\n\n");

        printf("<html><body>");
        printf("<h1>%s</h1>", array);
        printf("</body></html>");

        return 0;
}
```

POST example

Path relative:
"./cgi-bin/program_name"
Path absolute:
"http://www.cs.mcgill.ca/~user_name/cgi-bin/program_name"

Demo C

Parsing Special Characters

RFC 3986 section 2.2 *Reserved Characters* (January 2005)

!	*	'	(	)	;	:	@	&	=	+	$	,	/	?	#	[	]

RFC 3986 section 2.3 *Unreserved Characters* (January 2005)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

Reserved characters after percent-encoding

!	#	$	%	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Common characters after percent-encoding (ASCII or UTF-8 based)

newline	space	"	%	-	.	<	>	\	^	_	`	{	|	}	~	£	円
%0A *or* %0D *or* %0D%0A	%20	%22	%25	%2D	%2E	%3C	%3E	%5C	%5E	%5F	%60	%7B	%7C	%7D	%7E	%C2%A3	%E5%86%86

All ASCII or UTF-8 characters can be expressed this way: %XX

"bob smith@abc" → "bob+smith%40abc" → "bob smith@abc"

CGI program

parse unparse

Parsing Special Characters

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

int main() {
 char string[200], c, first, second;
 int a=0, result, n = atoi(getenv("CONTENT_LENGTH"));            // POST example

 while ((c = getchar()) != EOF && a<n)
 {
          if (a < n) {
                    if (c=='+') string[a]=' ';
                    else if (c=='%') {                           // %XX → c
                       first = getchar();
                       second= getchar();                        // dec = (16*first)+second

                       result = 0;
                       if (first>='A'&&first<='F') result = 16 * (10+first-'A');
                       else result=16*(first-'0');

                       if (second>='A'&&second<='F') result+=(10+second-'A');
                       else result += (second-'0');

                       string[a] = (char) result;
                    } else string[a]=c;

                    a++;
          }
 }

 string[a] = '\0';

 printf("Content-Type: text/html\n\n");  // notice before I print anything I print this
 printf("%s<br>",string);
 return 0;
}
```

input

output

SOCS and Python Programming

About webservers and the SOCS webserver

Contents

Python Programs for the web

- Already installed on mimi
- #!/usr/bin/python3
  - As first line of text file
- chmod +x
  - Like a Bash file
- Do not need to use .cgi file extension
- Do not need to place in cgi-bin, but please do so
- Runs with similar rules to C web programs

Python and CGI Input/output

```
#!/usr/bin/python3

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first_name = form.getvalue('first_name')
last_name  = form.getvalue('last_name')

print "Content-Type: text/html\n\n“
print "<html>"
print "<head>"
print "<title>Hello - Second CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s %s</h2>" % (first_name, last_name)
print "</body>"
print "</html>"
```

Same code works for both GET and POST

.cgi class

Python Output

```
#!/usr/bin/python3

print("ContentType: text/html\n\n")
print("hello world")
```

need the multiple carriage returns

```
#!/usr/bin/python3
print("Content-Type: text/html")    # HTML is following
print("")
print('<TITLE>CGI script output</TITLE>')
print('<H1>This is my first CGI script</H1>')
print('Hello, world!')
```

Will be rendered on the browser. Will overwrite the web page.

Python and Web

- Can run from any folder, but put in cgi-bin
- Use file extension .py or .cgi
- chmod +x
- Must begin with #!/usr/bin/python

```
#!/usr/bin/python3
print("Content-Type: text/html")    # HTML is following
print("")
print('<TITLE>CGI script output</TITLE>')
print('<H1>This is my first CGI script</H1>')
print('Hello, world!')
```

Demo

node + apache installed locally will fight w/ eachother

Final Comments

- Notice that we can do a lot without installing frameworks like Django.
- Same goes for all languages

Prepare for Next Class

- Assignments
  - How is mini 5 going?
- Labs
  - Lab B is being held this week
- Do on your own
  - Get your SOCS account working
    - Create a hello world homepage in HTML with 2 buttons
    - Button 1 calls a C (a.out) program and button 2 calls a Python script. These scripts overwrite the homepage outputting any sentence you like to the screen.