CS631 - Advanced Programming in the UNIX Environment

HTTP; Coding Practices

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann
jschauma@stevens.edu
https://www.cs.stevens-tech.edu/~jschauma/631/
HTTP: Hypertext

Information Management: A Proposal

Abstract

This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext system.

Keywords: Hypertext, Computer conferencing, Document retrieval, Information management, Project control

http://is.gd/JnZaN6
HTTP

Hypertext Transfer Protocol

RFC2616
HTTP

HTTP is a request/response protocol.
The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server

2. server responds
The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server
   - request method
   - URI
   - protocol version
   - request modifiers
   - client information

2. server responds
HTTP: A client request

$ telnet www.google.com 80
Trying 173.194.75.147...
Connected to www.google.com.
Escape character is '^[].'
GET / HTTP/1.0
The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server
   - request method
   - URI
   - protocol version
   - request modifiers
   - client information

2. server responds
   - status line (including success or error code)
   - server information
   - entity metainformation
   - content
HTTP: a server response

HTTP/1.0 200 OK
Date: Mon, 06 Nov 2017 18:53:47 GMT
Server: gws
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=116=W8F5wZJ58wbUGwVZIF1hm2Sr78_GGPunXZb56ka6k00YCFz1Lz7_y87DYcoO_I0n; expires=Tue, 08-May-2018 18:53:47 GMT; path=/; domain=.google.com; HttpOnly

<!doctype html><html itemscope="itemscope" itemtype="http://schema.org/WebPage">
<head><meta content="Search the..."
The Hypertext Transfer Protocol

Server status codes:

- **1xx** – Informational; Request received, continuing process
- **2xx** – Success; The action was successfully received, understood, and accepted
- **3xx** – Redirection; Further action must be taken in order to complete the request
- **4xx** – Client Error; The request contains bad syntax or cannot be fulfilled
- **5xx** – Server Error; The server failed to fulfill an apparently valid request
HTTP: A client request

$ telnet www.cs.stevens.edu 80
Trying 155.246.89.84...
Escape character is '^[']'.
GET / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Date: Mon, 06 Nov 2017 18:55:40 GMT
Server: Apache
Location: https://www.cs.stevens.edu/
Content-Length: 235
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head>
<body>

November 6, 2017
HTTP: A client request

```bash
[...]
GET / HTTP/1.1
Host: www.cs.stevens.edu

HTTP/1.1 302 Found
Date: Mon, 06 Nov 2017 19:02:10 GMT
Server: Apache
Location: https://www.stevens.edu/ses/cs
Vary: Accept-Encoding
Content-Length: 214
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```
HTTP: A client request

$ openssl s_client -crlf -servername www.stevens.edu -connect www.stevens.edu:443
[...]
GET / HTTP/1.1
Host: www.stevens.edu

HTTP/1.1 301 Moved Permanently
Date: Mon, 06 Nov 2017 19:09:21 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Set-Cookie: __cfduid=def9b13568803339571c6eff22b37f43c1509995361;
expires=Tue, 06-Nov-18 19:09:21 GMT; path=/; domain=.stevens.edu; HttpOnly
Last-Modified: Mon, 06 Nov 2017 13:27:15 GMT
Location: https://www.stevens.edu/schaefer-school-engineering-science/departments/computer-science
Via: 1.1 varnish-v4
Server: cloudflare-nginx
Strict-Transport-Security: max-age=15552000
HTTP: A client request

GET /schaefer-school-engineering-science/departments/computer-science HTTP/1.1
Host: www.stevens.edu

HTTP/1.1 200 OK
Date: Mon, 06 Nov 2017 19:11:20 GMT
Content-Type: text/html; charset=utf-8
Set-Cookie: __cfduid=dfa773452cb45ba58a5b356568cb3715e1509995480;
expires=Tue, 06-Nov-18 19:11:20 GMT; path=/; domain=.stevens.edu; HttpOnly
Last-Modified: Mon, 06 Nov 2017 15:44:53 GMT
Via: 1.1 varnish-v4
X-Generator: Drupal 7 (http://drupal.org)
X-Varnish: 12117551 14890606
Strict-Transport-Security: max-age=15552000
Server: cloudflare-nginx

<!DOCTYPE html>
<html lang="en" class="no-js"
HTTP: A client request
HTTP - more than just text

HTTP is a *Transfer Protocol* – serving *data*, not any specific text format.

- **Accept-Encoding** client header can specify different formats such as *gzip*, *Shared Dictionary Compression over HTTP (SDCH)* etc.
- **corresponding server headers**: *Content-Type* and *Content-Encoding*
HTTP - more than just static data

HTTP is a *Transfer Protocol* – what is transferred need not be static; resources may generate different data to return based on many variables.

- CGI – resource is *executed*, needs to generate appropriate response headers
- server-side scripting (ASP, PHP, Perl, ...)
- client-side scripting (JavaScript/ECMAScript/JScript,...)
- applications based on HTTP, using:
  - AJAX
  - RESTful services
  - JSON, XML, YAML to represent state and abstract information
HTTP in your final project

- protocol versions supported: 1.0
- request methods supported: GET, HEAD
- request headers supported: If-Modified-Since
- response headers included: Date, Server, Last-Modified, Content-Type, Content-Length
HTTP in your final project

- accept connections, read input from client
- timeout idle connections
- parse and validate input
- generate proper HTTP codes
- log connection information
- send headers
- send response
- close connection
HTTP in your final project

Client data parsing:

```
method uri protocol
<optional headers>
```

- `method` not in GET, HEAD? => 400
- protocol != "HTTP/1.0"? => 505
- uri too long? => 400
HTTP in your final project

URI processing:
- begins with "~/something"? => userdir handling
- begins with "/cgi-bin/"? => cgi handling
- avoid breaking out of docroot via "../.." etc.
- ends in "/"? => look for 'index.html' or generate index on the fly
HTTP in your final project

https://www.cs.stevens.edu/~jschauma/631/test-sws.sh
Coding Practices

Beyond the Unix Philosophy
The Unix Philosophy

This is the Unix philosophy:

Write programs that do one thing and do it well.

Write programs to work together.

Write programs to handle text streams, because that is a universal interface.
Unix basics

Write your code and tools such that they work well within the Unix ecosystem:

- write portable code, target different Unix flavors
- use `strerror(3)/perror(3)`
- errors go to `stderr`
- use meaningful return codes
- follow Unix conventions when using e.g. flags, files, config files, passwords, environment variables, ...
Unix basics

Not like this:

```
$ hStub fRumbl
Something error ocured!
$ echo $? 0
```

But like this:

```
$ hStub fRumbl
Unable to open file: '/usr/share/hStub/fRumbl': No such file or directory
$ echo $? 1
```
The Zen of Python

Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren’t special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you’re Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it’s a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let’s do more of those!
Readability counts

Visual flow:

- use spaces/tabs/indentation *consistently*
- use a standard width terminal (~80 chars)
- refactor if code wraps / trails off right side
- refactor if logic doesn’t fit into about one screen height
- never repeat the same code block
Readability counts

Not like this:

```c
if (smCndt) {
    foo = fncSmthng();
    if (strcmp(foo->bar, "MUBLEFRUMBLE") { 
        if (otherWat
&& frobbleEnabled) {
            printf("Something error ocured!\n")
            exit(0);
        }
    }
    else {
        frbF();
    }
}
```
Readability counts

But like this:

```c
void handleFoo(foo_t *foo) {
    if (strncmp(foo->bar, "MUMBLEFRUMBLE", strlen("MUMBLEFRUMBLE")) > 0) {
        if (otherWat && frobbleEnabled) {
            fprintf(stderr, "Unexpected value '%%s'.", foo->bar);
        }
    }
}

[...]

if (condition) {
    if ((foo = createFoo()) != NULL) {
        handleFoo(foo);
    } else {
        frobFoo(foo);
    }
}
```
Readability counts

Code is language:

- you are not charged per character
- use *descriptive* function and variable names
- use comments *where necessary*; explain *why*, not *what*
- don’t use magic numbers
- write boring code
Structure

- "do one thing and do it well" also applies to functions
- eliminate side-effects
- minimize the use of global variables
- keep `open(2)/close(2), malloc(3)/free(3), etc. in same` (visual/logical) scope
- separating code into multiple files helps clarify your interfaces
Pitfalls

- check the return value of any function that can fail!
- avoid file I/O whenever possible
- avoid using temporary files whenever possible
- don’t assume you can write to the current working directory
- be explicit in setting permissions; set/use `umask(2)`
- use an exit handler to clean up after yourself
- retain idempotency whenever possible
Never trust the user

- your tool must be safe even if hostile input is given
- never trust the environment
- sanitize and validate all input
- you can’t exhaustively identify all “bad” cases, so use a whitelist, not a blacklist
How to figure things out

- know your editor (tags, folds, jumping, ...)
- use a debugger
- use the source
- use `strace(1)/dtrace(1)/ktrace(1)` etc.
- write separate code to prove yourself right (or wrong)
Reading

HTTP etc.:

- RFC 2616, 2818, 3875
- http://httpd.apache.org/docs/
- http://www.w3.org/Protocols/
- REST: http://is.gd/leSvGa

Coding practices:

- https://is.gd/xIf0gI
- https://is.gd/nXh3Pq