OS 2018

Homework2:

Simple myhttp server

(Due date: 2018/11/22 23:59)

Objectives

- Understand HTTP client-server model
- Understand multi-thread programming

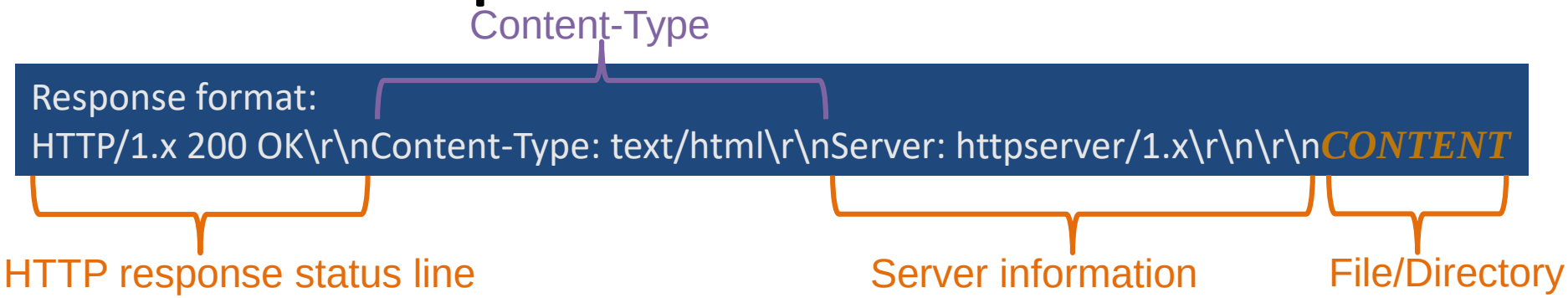
Requirements

- A Simple Http-like Server
- A Simple Http-like Client
- Use Internet sockets for client-server communication
- Use thread pool to handle client requests
  - Pthreads
  - One type of requests: GET

Requirements for the Server

- Use a simple http-like protocol, 1.x(not real http protocol)
  - Handle one type of requests: GET
  - Use internet socket to connect with clients
  - Follow the response format, header+content (p.5, p.9)
- Support multithreading
  - Must implement thread pool and maintain request queue
    - Main thread will *accept* for any new connections and put the request into a request queue
    - The other threads keep trying to take requests from the request queue and handle the request
  - lock is required for request queue accesses
- Usage:
  - $ ./myhttpserver –r *root* -p *port* -n *thread_number*

Server Response Format



- HTTP header lines–
  - HTTP response status line: HTTP protocol version, status code and description
    - HTTP protocol version: HTTP/1.x (fixed string)
    - Status code: 200 (You should support all the status codes *in the next slide*)
    - Description: OK\r\n
  - Content-Type:
    - For file, “Content-Type: ”+ filetype + ”\r\n” (You should support all the *file types in the next slide*)
    - For directory, “Content-Type: directory\r\n”
  - Server information: (fixed string)
- File/Directory content:
  - For file: file content to header
  - For directory: names of the files and subdirectories in that directory (except “.” and “..”). Names are separated by blank space (similar to ls)

Supported File Types & Status Codes

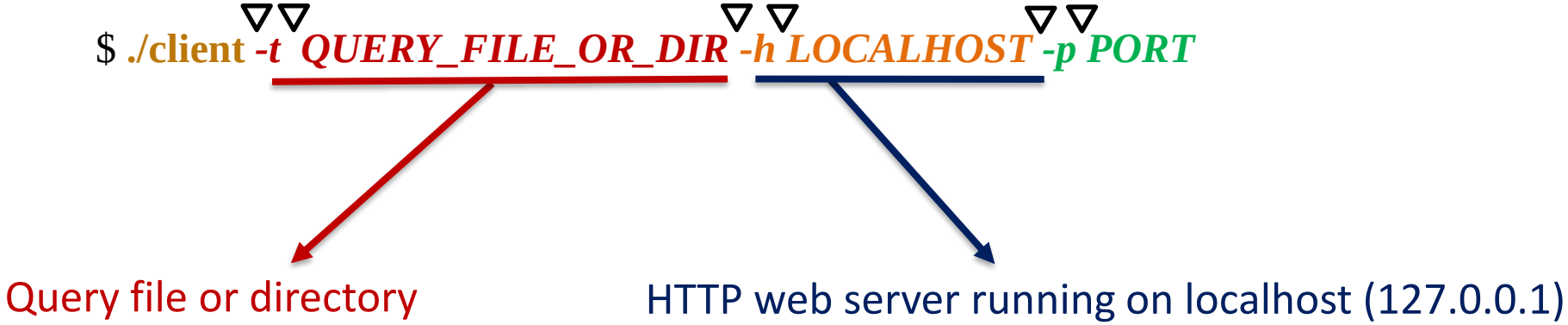
File extensions

File type

Status code

```
32  extn extensions[] = {
33      {"htm",  "text/html"},
34      {"html", "text/html"},
35      {"css",  "text/css"},
36      {"h",  "text/x-h"},
37      {"hh",  "text/x-h"},
38      {"c",  "text/x-c"},
39      {"cc",  "text/x-c"},
40      {"json", "application/json"},
41      {0, 0}
42  };
43
44  enum {
45      OK = 0,
46      BAD_REQUEST,
47      NOT_FOUND,
48      METHOD_NOT_ALLOWED,
49      UNSUPPORT_MEDIA_TYPE
50  };
51
52  const int status_code[] = {
53      200, /* OK */
54      400, /* Bad Request */
55      404, /* Not Found */
56      405, /* Method Not Allowed */
57      415, /* Unsupported Media Type */
58  };
```

Requirements for the Client



The format of a simple my_HTTP request is:
"GET *QUERY_FILE_OR_DIR* HTTP/1.x\r\nHOST: *LOCALHOST*:*PORT* \r\n\r\n"

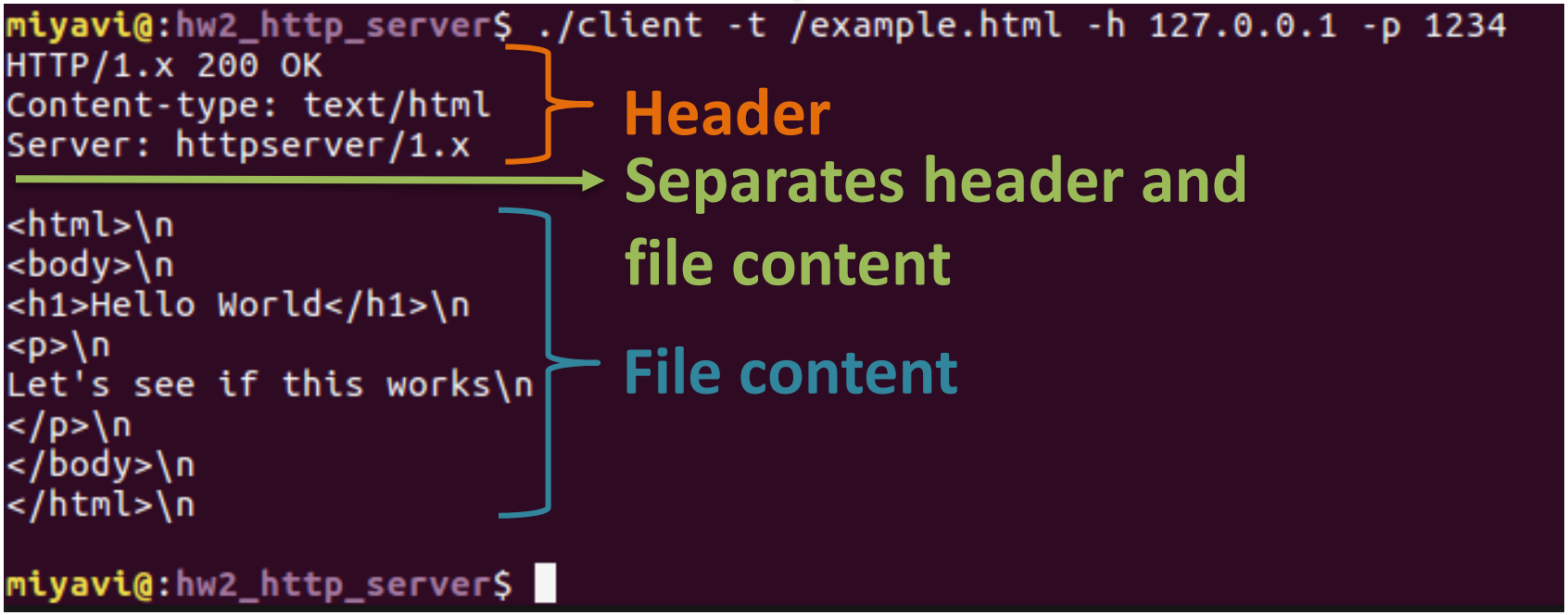
- Size limit:
  - Maximum size of the ***QUERY_FILE_OR_DIR***: **128 bytes**

Requirements for the Client(cont.)

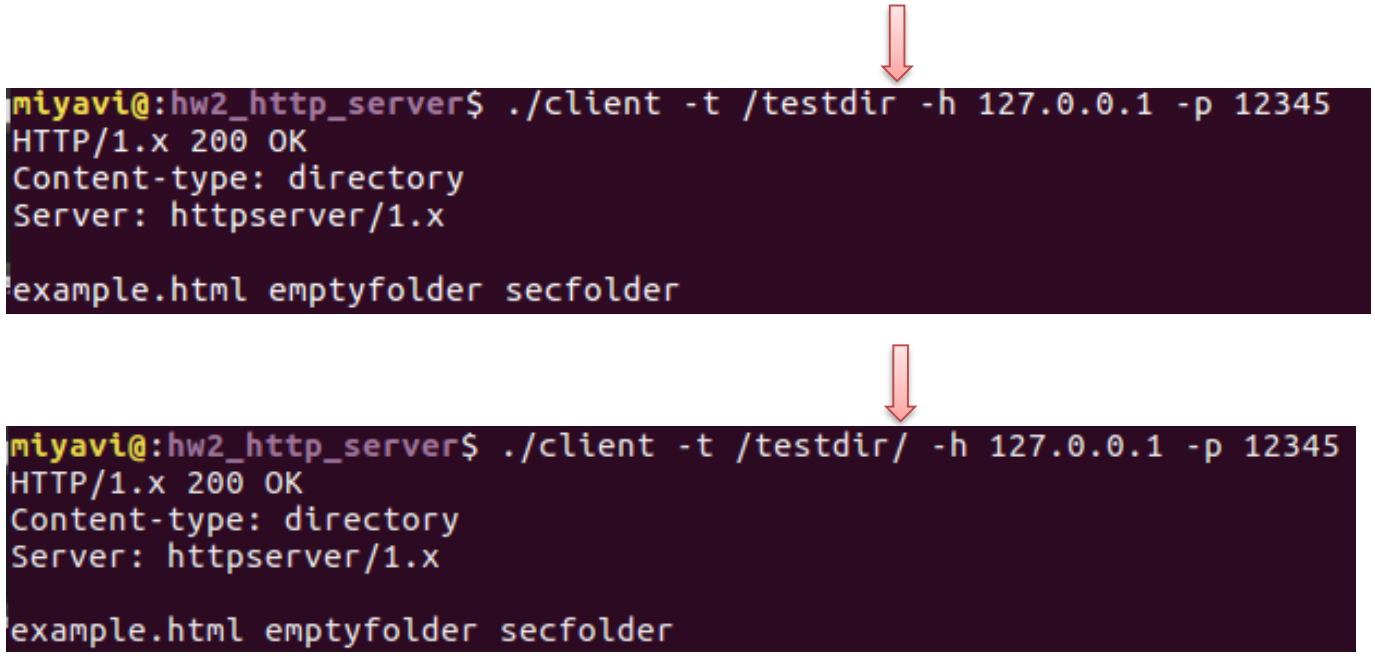
- Print out the header+file or directory content.
  - Example: p.9-p.16
- For each file request, save the file content under the client's ./output directory
  - Need to maintain the same directory hierarchy as server
  - Example: p.12
- If the content type is `*directory*`, create a thread for each file/subdirectory in that directory to get the content of the file/subdirectory (p.11)

Example (1)

Must start with slash



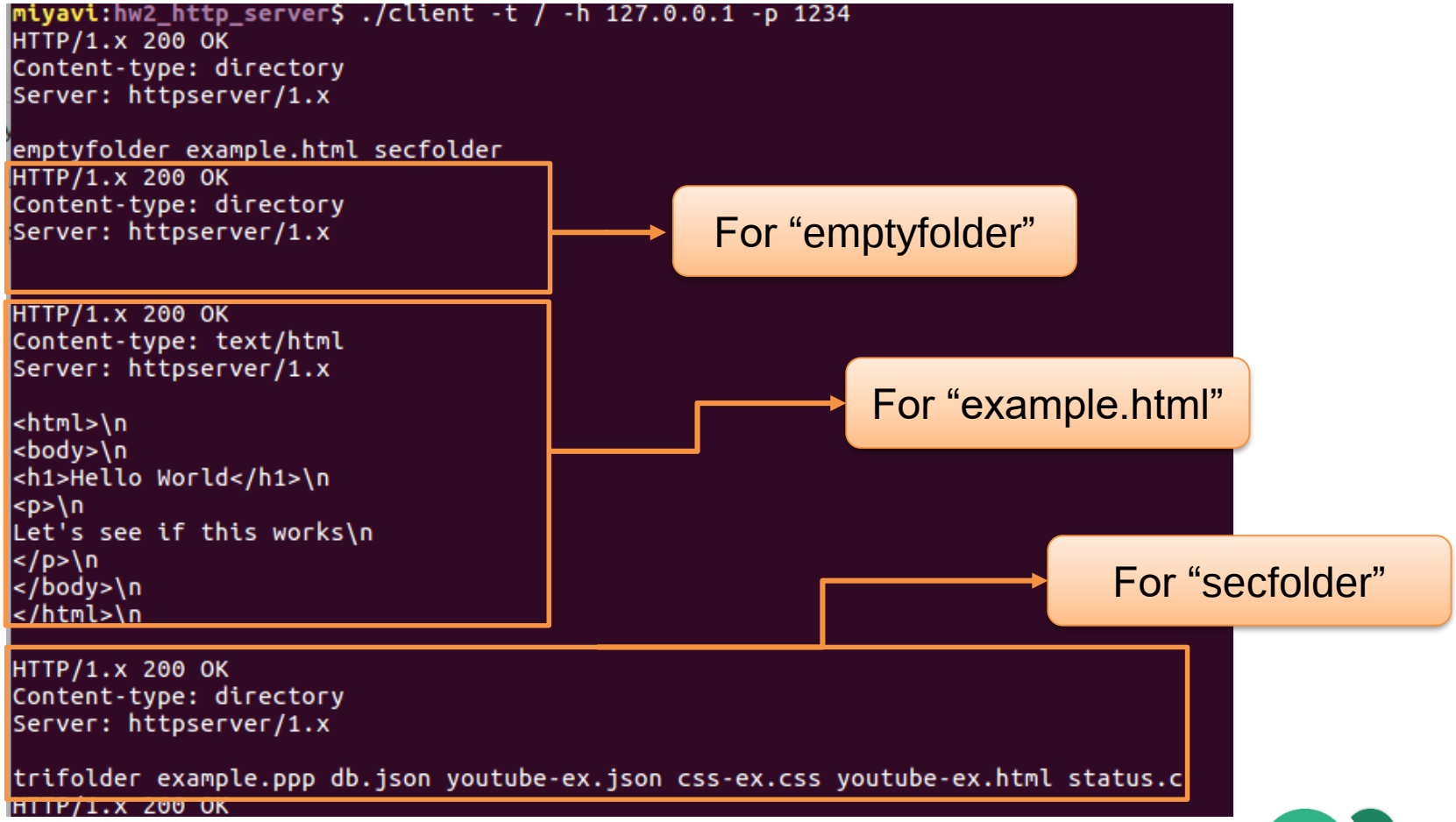
Example (2)



- **Do NOT assume that a directory request will have trailing slash in the query string.**
- **For example.html, emptyfolder and secfolder shown above, create a thread to generate a request to server for each of the subdir/file.**

Example (3)

- For a directory request, create a thread to generate a new request to the server for each subdir/file.



Example (4)

- When saving file content, create subdirectories if needed to maintain the same directory hierarchy with the server.

```
miyavi@:hw2_http_server$ ./client -t /secfolder/trifolder/db.json -h 127.0.0.1 -p 1234
HTTP/1.x 200 OK
Content-type: application/json
Server: httpserver/1.x

{
  "clients": [
    {
      "id": "59761c23b30d971669fb42ff",
```

Directories not exist under “output”

Result:

```
miyavi@:output$ ls ./secfolder/trifolder/
db.json
miyavi@:output$
```

Example (5) – Error Conditions(1)

- ***QUERY_FILE_OR_DIR*** in request doesn't start with a slash.
- Status code: 400
- Content-type: empty
- Status description: BAD_REQUEST

```
miyavi@:hw2_http_server$ ./client -t example.html -h 127.0.0.1 -p 1234
HTTP/1.x 400 BAD_REQUEST
Content-type:
Server: httpserver/1.x


miyavi@:hw2_http_server$
```

Example (6) – Error Conditions(2)

- **No such file or directory**
- Status code: 404
- Content-type: empty
- Status description: NOT_FOUND

```
miyavi@:hw2_http_server$ ./client -t /noexist.html -h 127.0.0.1 -p 1234
HTTP/1.x 404 NOT_FOUND
Content-type:
Server: httpserver/1.x


miyavi@:hw2_http_server$
```

Example (7) – Error Conditions(3)

- **Unsupported file types**
- Status code: 415
- Content-type: empty
- Status description: UNSUPPORT_MEDIA_TYPE

```
miyavi@:hw2_http_server$ ./client -t /example.ppp -h 127.0.0.1 -p 1234
HTTP/1.x 415 UNSUPPORT_MEDIA_TYPE
Content-type:
Server: httpserver/1.x


miyavi@:hw2_http_server$
```

Example (8) – Error Conditions(4)

- Unsupported Method
  - Support “GET” only
  - Others like POST/HEAD/get is not allowed
- Status code: 405
- Content-type: empty
- Status description: METHOD_NOT_ALLOWED

The format of request message send to server:
“get */testfolder* HTTP/1.x\r\nHOST: *127.0.0.1:1234*\r\n\r\n"

```
miyavi@:hw2_http_server$ ./client -t /testfolder -h 127.0.0.1 -p 1234
HTTP/1.x 405 METHOD_NOT_ALLOWED
Content-type:
Server: httpserver/1.x


miyavi@:hw2_http_server$
```

References

- Manual Page
  - Thread_pool
  - pthreads
  - pthread_mutex_lock, pthread_mutex_init
  - pthread_spin_lock , pthread_spin_init
  - socket
  - sem_overview
- HTTP
  - http_introduction