

Network Programming

Annalise Tarhan

October 11, 2020

11 Homework Problems

11.6

11.6.1 Modify TINY so that it echoes every request line and request header.

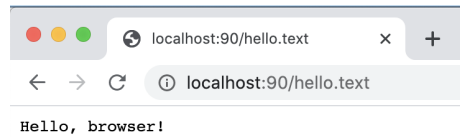
Instead of calling `serve_static` or `serve_dynamic`, `doit` calls `serve_echo`:

```
void serve_echo(int fd)
{
    char buf[MAXLINE];

    rio_t rio;
    rio_readinitb(&rio, fd);

    while(strcmp(buf, "\r\n")) {
        rio_readlineb(&rio, buf, MAXLINE);
        rio_writen(fd, buf, strlen(buf));
    }
}
```

**11.6.2 Use a browser to make a request to TINY for static content.
Capture the output from TINY in a file.**



```
void save_to_file(void *usrbuf, size_t n)
{
    int fd = open(
        "/Users/annalisetarhan/Desktop/output.txt",
        O_CREAT|O_APPEND|O_WRONLY, 0);
    rio_writen(fd, usrbuf, n);
    close(fd);
    return;
}
```

```
void serve_static(int fd, char *filename, int filesize)
{
    ...
    rio_writen(fd, srcp, filesize);
    save_to_file(srcp, filesize);
    munmap(srcp, filesize);
    ...
}
```

**11.6.3 Inspect the output from TINY to determine the version of
HTTP your browser uses.**

Chrome: HTTP/1.0

11.6.4 Determine the meaning of each header in the HTTP request.

Accept: text/html, application/xhtml+xml ...

List of media ranges which are acceptable as a response to the request.

Accept-Encoding: gzip, deflate, br

Similar to Accept, but restricts the content-coding values which are acceptable in the response.

Accept-Language: en-US,en;q=0.9

Similar to accept, but restricts the set of natural languages that are preferred

as a response to the request.

Connection: keep-alive

Allows the sender to specify options that are desired for that particular connection.

Host: localhost:90

Specifies the internet host and port number of the resource being requested.

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)

Contains information about the user agent originating the request.

11.7 Extend TINY so that it serves MPG video files.

```
void get_filetype(char *filename, char *filetype)
{
    if (strstr(filename, ".html"))
        strcpy(filetype, "text/html");
    ...
    else if (strstr(filename, ".mpg"))
        strcpy(filetype, "video/mpeg");
    ...
}
```

11.8 Modify TINY so that it reaps CGI children inside a SIGCHLD handler instead of explicitly waiting for them to terminate.

```
void sigchld_handler(int s)
{
    int olderrno = errno;
    waitpid(-1, NULL, 0);
    errno = olderrno;
}
```

```
int main(int argc, char **argv)
{
    ...
    signal(SIGCHLD, sigchld_handler);
    ...
}
```

11.9 Modify TINY so that when it serves static content, it copies the requested file to the connected descriptor using `malloc`, `rio_readn`, and `rio_writen`, instead of `mmap` and `rio_writen`.

```
void serve_static(int fd, char *filename, int filesize)
{
    int srcfd;
    char *srcp;
    ...
    srcfd = open(filename, ORDONLY, 0);
    srcp = malloc(filesize);
    rio_readn(srcfd, srcp, filesize);
    rio_writen(fd, srcp, filesize);
    free(srcp);
}
```

11.10

11.10.1 Write an HTML form for the CGI adder function.

```
<!DOCTYPE html>
<html>
<head>
<form action="/cgi-bin/adder" method="GET">
    <label for="addend">Addend:</label><br>
    <input type="number" id="addend" name="addend"><br>
    <label for="augend">Augend:</label><br>
    <input type="number" id="augend" name="augend"><br>
    <input type="submit" value="Submit">
</form>
</head>
</html>
```

```
void serve_dynamic(int fd, char *filename, char *cgiargs)
{
    ...
    char stripped_args[MAXLINE], *ptr1, *ptr2, *ptr3;

    ptr1 = index(cgiargs, '=');
    // Hide the first = to find the second
    *ptr1 = ' ';
    ptr2 = index(cgiargs, '=');
    ptr3 = index(cgiargs, '&');
```

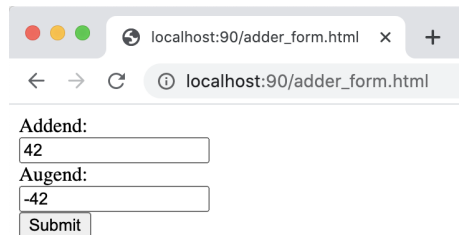
```

// Null terminate after first argument
*ptr3 = '\0';
strcpy(stripped_args, ptr1+1);
strcat(stripped_args, "&");
strcat(stripped_args, ptr2+1);

if (fork() == 0) {
    setenv("QUERY_STRING", stripped_args, 1);
    dup2(fd, STDOUT_FILENO);
    execve(filename, emptylist, environ);
}
}

```

11.10.2 Check your work by using a real browser to request the form from TINY, and then display the dynamic content generated by adder.

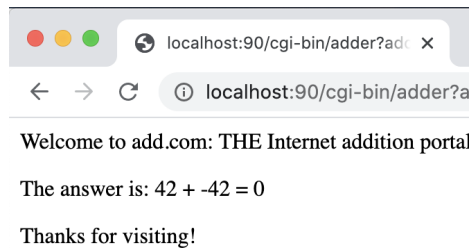


localhost:90/adder_form.html

← → ↻ ⓘ localhost:90/adder_form.html

Addend:

Augend:



localhost:90/cgi-bin/adder?adc

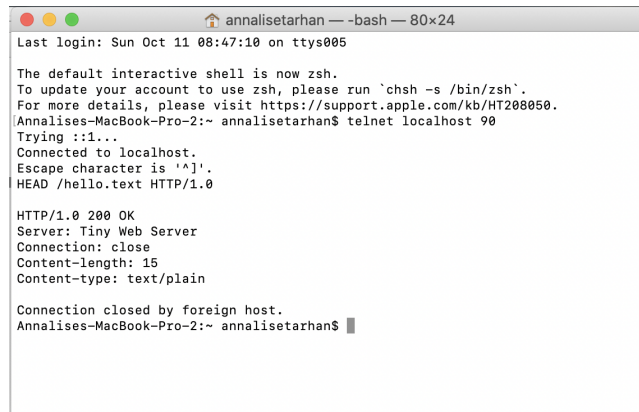
← → ↻ ⓘ localhost:90/cgi-bin/adder?a

Welcome to add.com: THE Internet addition portal

The answer is: 42 + -42 = 0

Thanks for visiting!

11.11 Extend TINY to support the HTTP HEAD method. Check your work using TELNET.



```
annalisetarhan ~ -bash — 80x24
Last login: Sun Oct 11 08:47:10 on ttys005

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Annalises-MacBook-Pro-2:~ annalisetarhan$ telnet localhost 90
Trying ::1...
Connected to localhost.
Escape character is '^J'.
HEAD /hello.text HTTP/1.0

HTTP/1.0 200 OK
Server: Tiny Web Server
Connection: close
Content-length: 15
Content-type: text/plain

Connection closed by foreign host.
Annalises-MacBook-Pro-2:~ annalisetarhan$
```

```
void doit(int fd)
{
    ...
    if (strcasecmp(method, "GET") &&
        strcasecmp(method, "HEAD")) {
        clienterror(fd, method, "501",
            "Not implemented",
            "Tiny does not implement this method");
        return;
    }

    int headers_only = (!strcasecmp(method, "HEAD"));
    ...
    if (is_static) {
        if (!(S_ISREG(sbuf.st_mode)) ||
            !(S_IRUSR & sbuf.st_mode)) {
            clienterror(fd, filename, "403",
                "Forbidden",
                "Tiny couldn't read the file");
            return;
        }
        serve_static(fd, filename, sbuf.st_size,
            headers_only);
    }
    else {
        if (!(S_ISREG(sbuf.st_mode)) ||
            !(S_IXUSR & sbuf.st_mode)) {
            clienterror(fd, filename, "403",
```

```

        "Forbidden",
        "Tiny couldn't run the program");
        return;
    }
    serve_dynamic(fd, filename, cgiargs,
        headers_only);
}

```

```

void serve_dynamic(int fd, char *filename, char *cgiargs,
int headers_only)
{
    char buf[MAXLINE], *emptylist[] = { NULL };

    sprintf(buf, "HTTP/1.0 200 OK\r\n");
    rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Server: Tiny Web Server\r\n");
    rio_writen(fd, buf, strlen(buf));

    if (headers_only) return;
    ...
}

```

```

void serve_static(int fd, char *filename, int filesize,
int headers_only)
{
    int srcfd;
    char *srp, filetype[MAXLINE], buf[MAXBUF];

    get_filetype(filename, filetype);
    sprintf(buf, "HTTP/1.0 200 OK\r\n");
    sprintf(buf,
        "%sServer: Tiny Web Server\r\n", buf);
    sprintf(buf, "%sConnection: close\r\n", buf);
    sprintf(buf, "%sContent-length: %d\r\n", buf, filesize);
    sprintf(buf, "%sContent-type: %s\r\n\r\n", buf, filetype);
    rio_writen(fd, buf, strlen(buf));
    printf("Response headers:\n");
    printf("%s", buf);

    if (headers_only) return;
    ...
}

```

11.12 Extend TINY so that it serves dynamic content requested by the HTTP POST method.

```
void doit(int fd)
{
    int is_static, is_head, is_post;
    ...
    if (strcasecmp(method, "GET") &&
        strcasecmp(method, "HEAD") &&
        strcasecmp(method, "POST")) {
        clienterror(fd, method, "501",
            "Not implemented",
            "Tiny does not implement this method");
        return;
    }

    is_head = !strcasecmp(method, "HEAD");
    is_post = !strcasecmp(method, "POST");

    is_static = parse_uri(uri, filename, cgiargs) &&
        strcasecmp(method, "POST");
    ...
    if (is_post) {
        rio_read(&rio, cgiargs, MAXBUF);
    }
    ...
}
```


11.13 Modify TINY so that it deals cleanly (without terminating) with the SIGPIPE signals and EPIPE errors that occur when the write function attempts to write to a prematurely closed connection.

```
sigjmp_buf buf;

void handler(int sig)
{
    siglongjmp(buf, 1);
}

int main(int argc, char **argv)
{
    ...
    while (1) {
        if (!sigsetjmp(buf, 1)) {
            signal(SIGPIPE, handler);
            signal(EPIPE, handler);
        }
        ...
    }
}
```