Computer networks - 1DV701

Assignment 2

Author: Michael Johansson & Jacob Heyder

Semester: VT 2017

Contents

1 Assignment summary **1**

2 Problem one **1**

3 Problem two **3**
3.1 VG.1 HTTP status code implementation 4
3.2 VG.2 POST vs PUT 6

4 Problem three **7**

1 Assignment summary

2 Problem one

We choose to create an Index file every time an user requests a directory. This way we have index links to the files in that directory to help with the navigation and it looks good. But if the user ask for a file/directory that doesn't exist on the server we send back a 404(File not found).We also link to our secret folder in this main index but if we click on it we get an 403 (access denied).

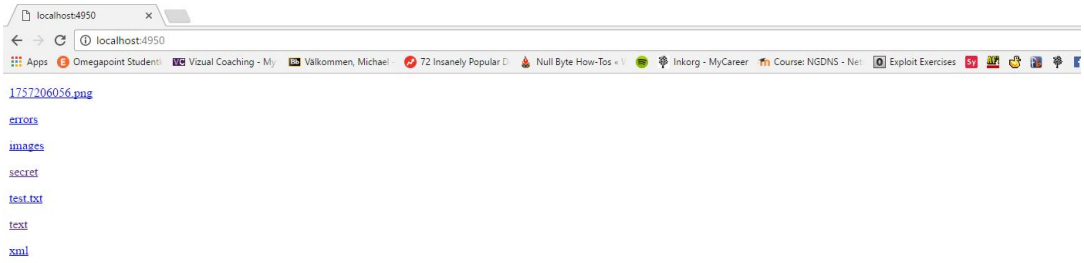

Figure 2.1: GET request on a directory, responds with index.html

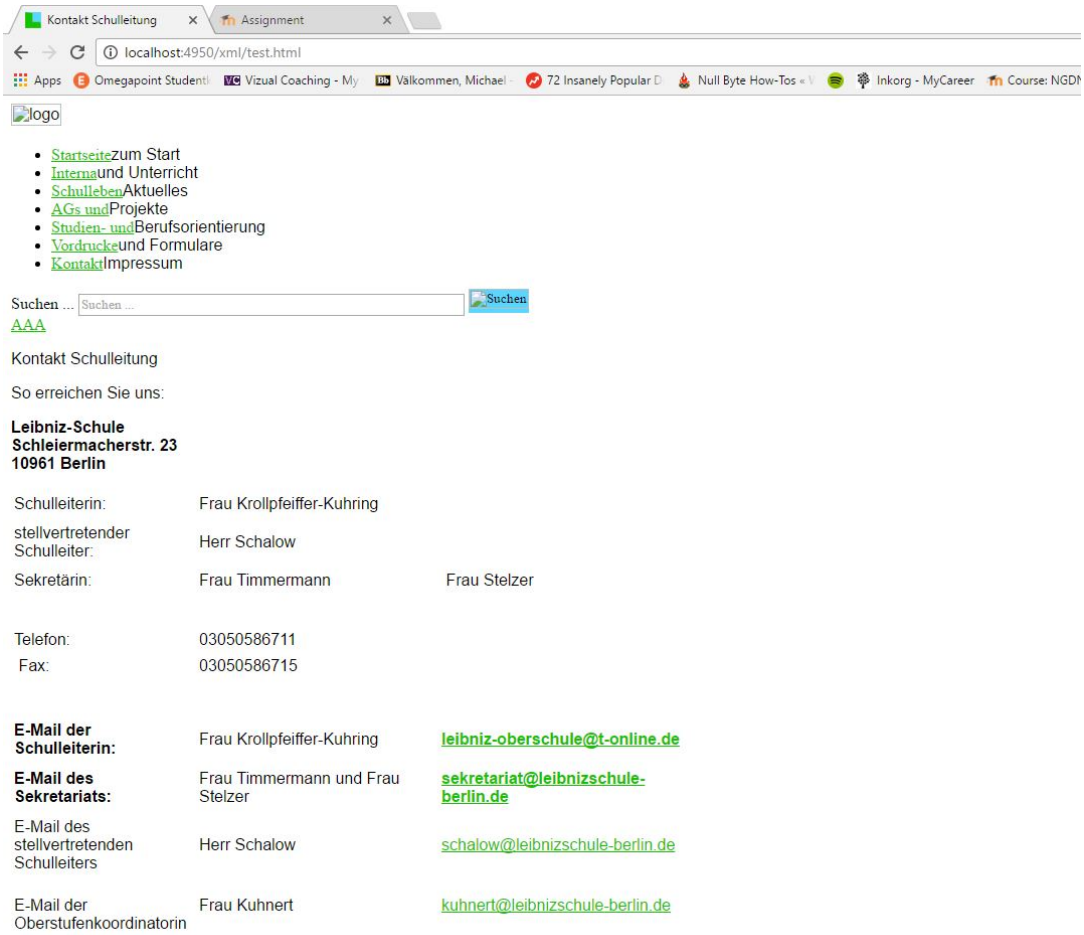

Figure 2.2: GET request on a HTML file

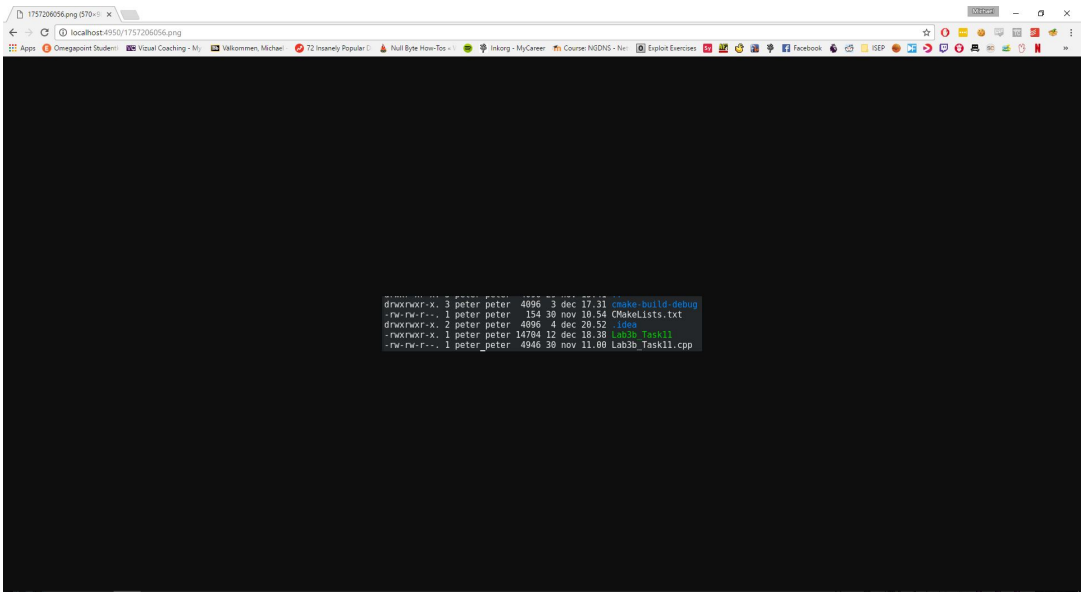

Figure 2.3: GET request on an image

3 Problem two

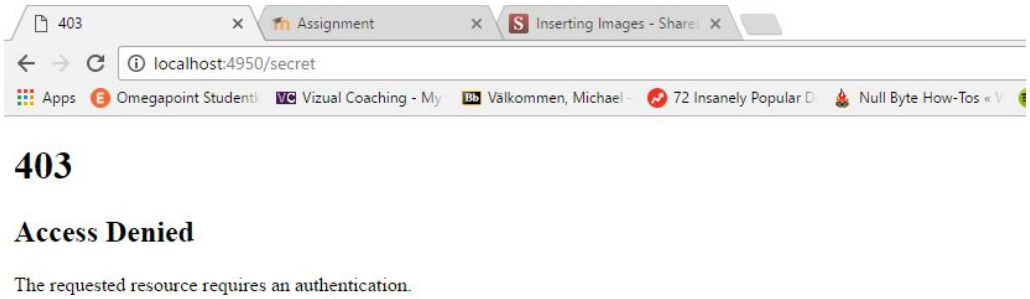

Figure 3.1: 403 Access denied when trying to access our secret folder

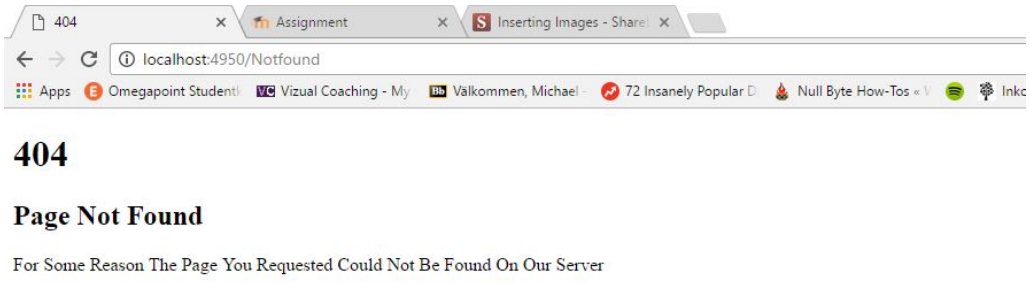

Figure 3.2: 404 Not Found when trying to GET a resource that don't exist on the server

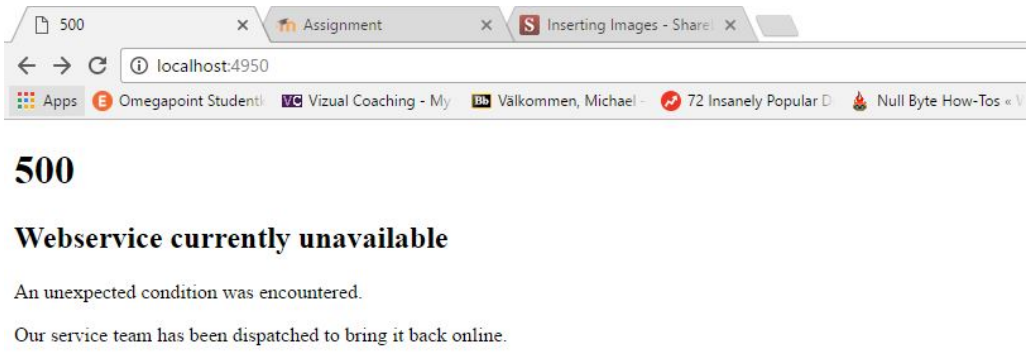

Figure 3.3: 500 for any internal server errors, socket error etc.

3.1 VG.1 HTTP status code implementation

For some images below we just show the error.html response but we send the status code as well for every error.

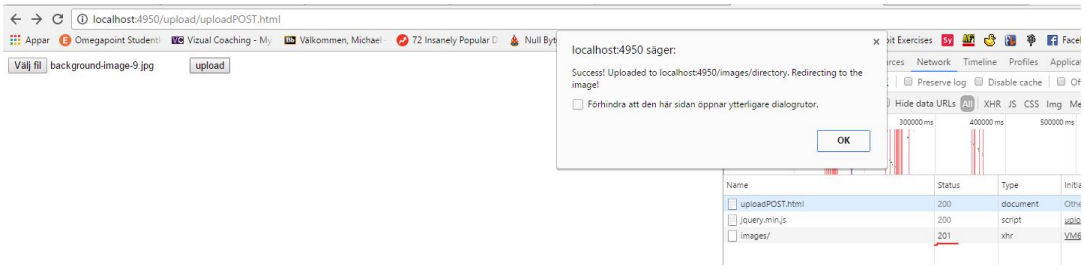

Figure 3.4: 201 when the user creates a new resource with POST/PUT.

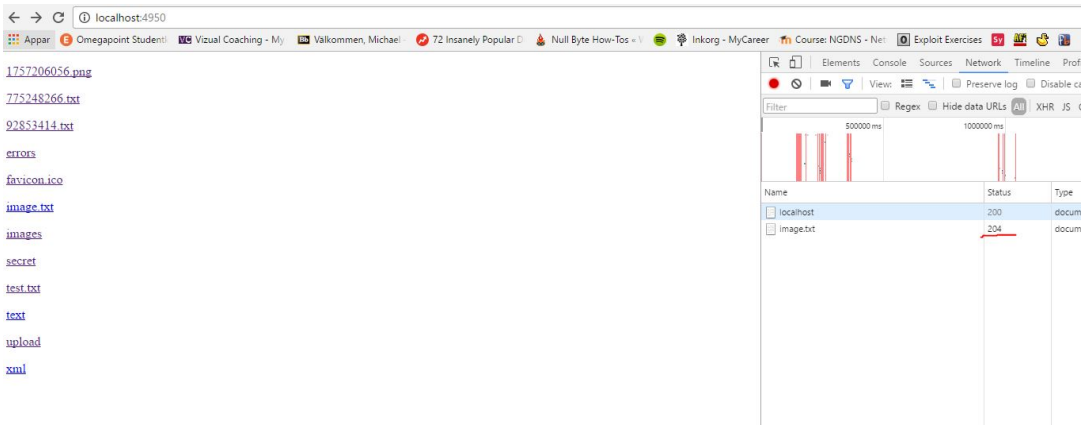

Figure 3.5: 204 if the user tries to GET a empty file

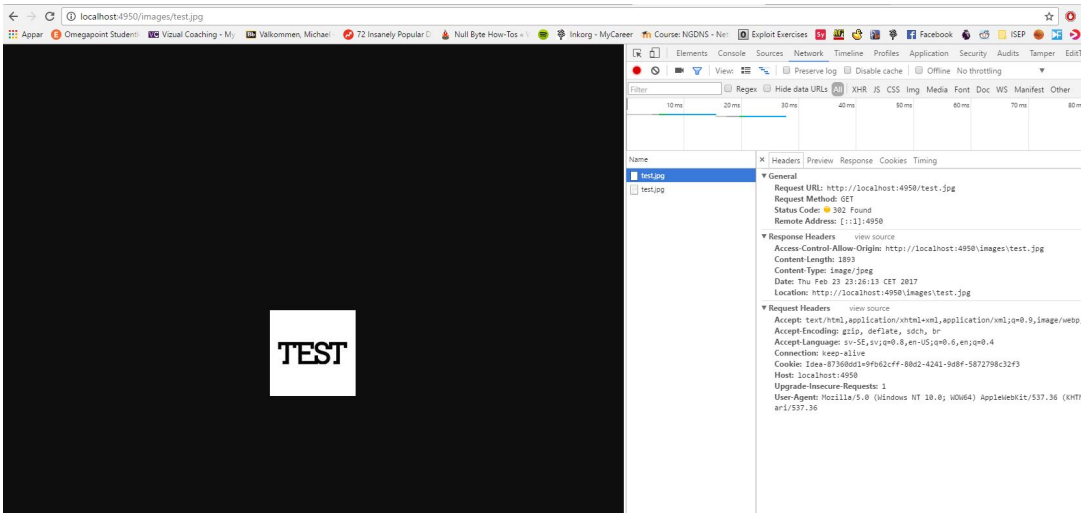

Figure 3.6: 302 if we have specified that a specific file has changed location on the server, redirect the browser to the new location with the location header.

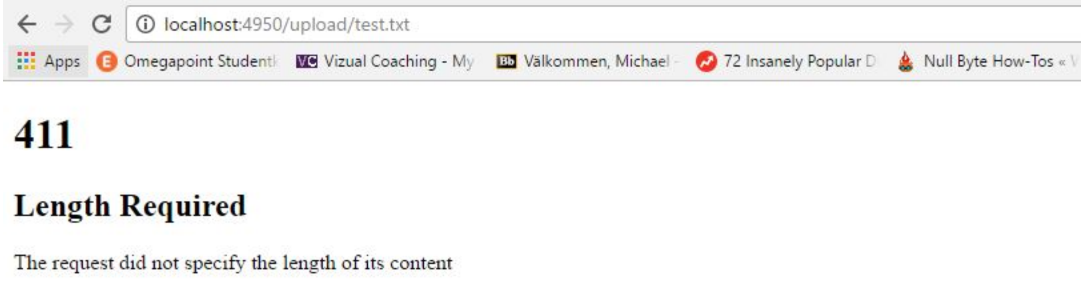

Figure 3.7: 411 if the POST/PUT body is empty

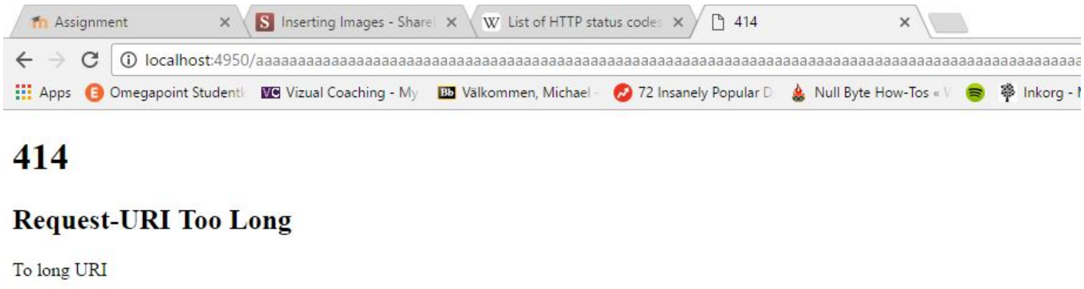

Figure 3.8: 414 if the URI is longer then 2048 characters.

Figure 3.9: // CHECK THIS ONE

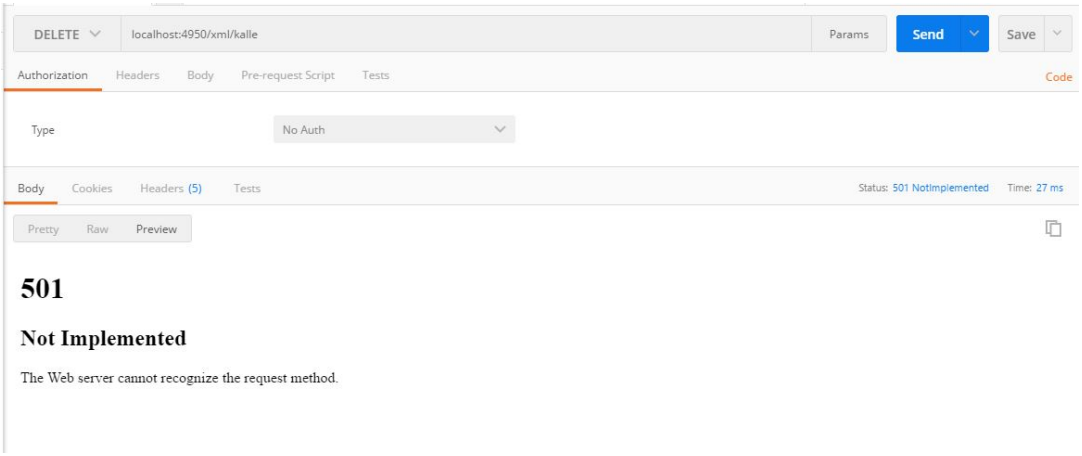

Figure 3.10: 501 for any HTTP methods that we haven't implemented, like in this case the DELETE method.

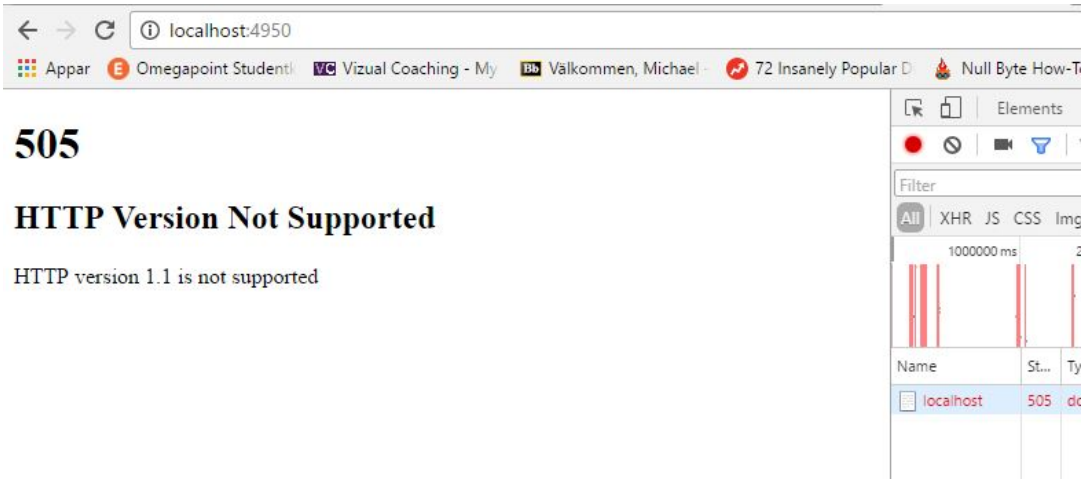

Figure 3.11: 505 for HTTP version 1.1. We support all HTTP versions when we run the server, but we can set this exception if we don't want to support a special HTTP version later on.

3.2 VG.2 POST vs PUT

We use POST when the user uploads a new resource that our sever handles where it should be placed. In our POST implementation the user can upload an image through our upload page. The server then creates that images with a hash as the file name and under the images folder.

If instead the user has the exact request-URI then we can use PUT to create or overwrite the file on that exact URI.

So in short the POST is used for when the client don't need to know the exact URI and just upload the file where we want it. The PUT when the client want to Create or replace a file on an exact URI.

4 Problem three

Using Telnet we are creating a raw TCP connection that sends and receive plain text. So as we se in figure 4.1 for our request for a text file we get the text as is. But if we then ask for an image like in figure 4.2 we get the image information as text as well. So the Telnet interprets the image binary as plain text and that's why we get those weird symbols.

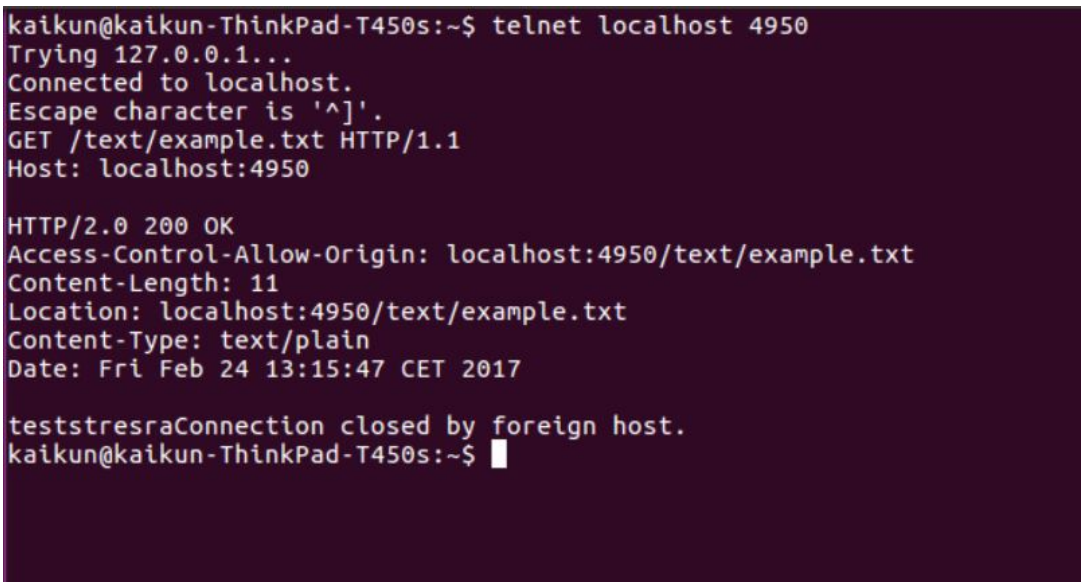

Figure 4.1: Telnet client asking for a .txt file, reads it in plain text

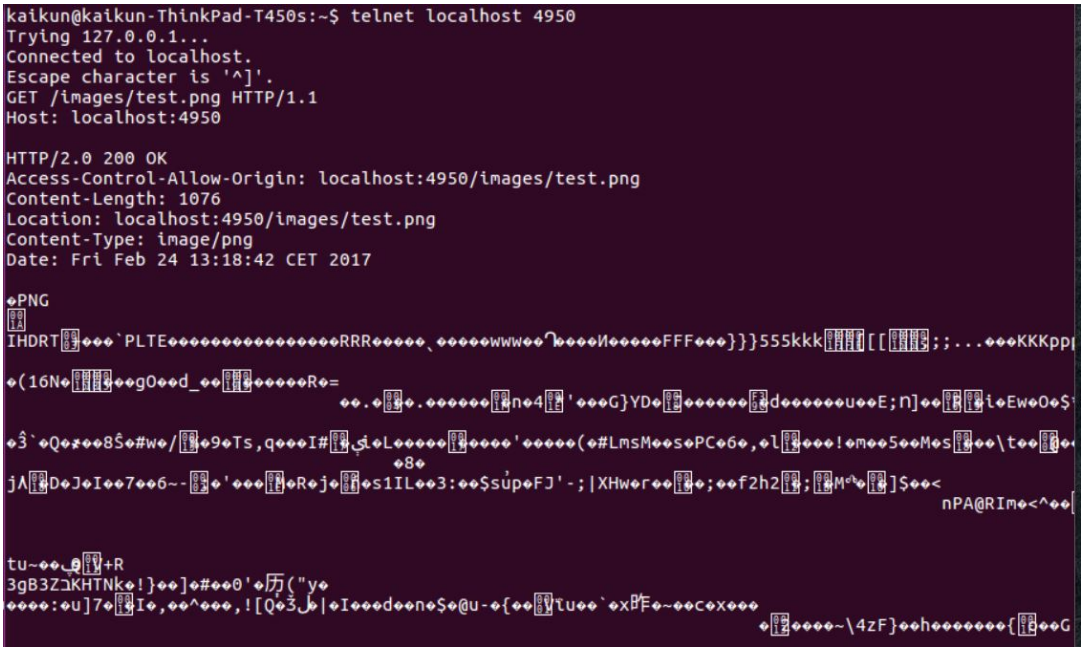

Figure 4.2: Telnet client asking for a .PNG file, also reads it as plain text