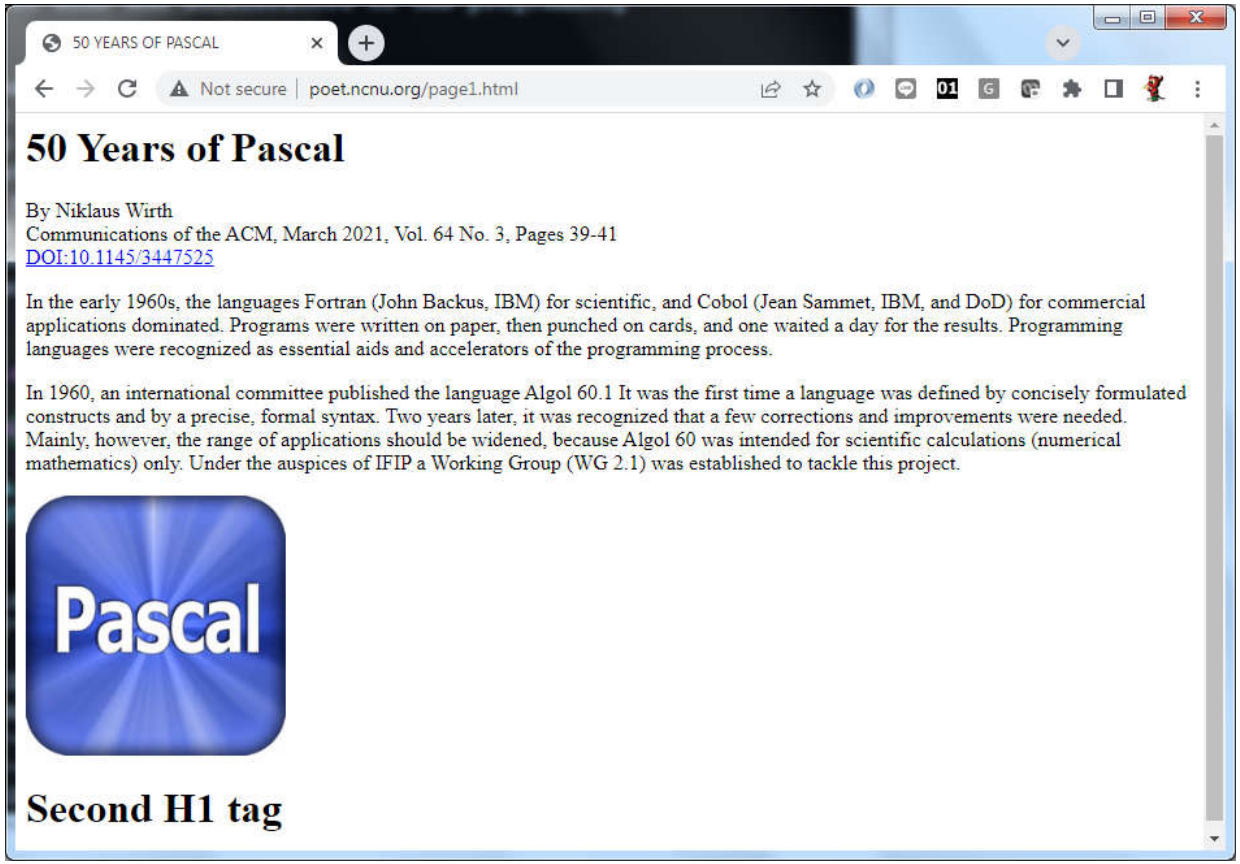
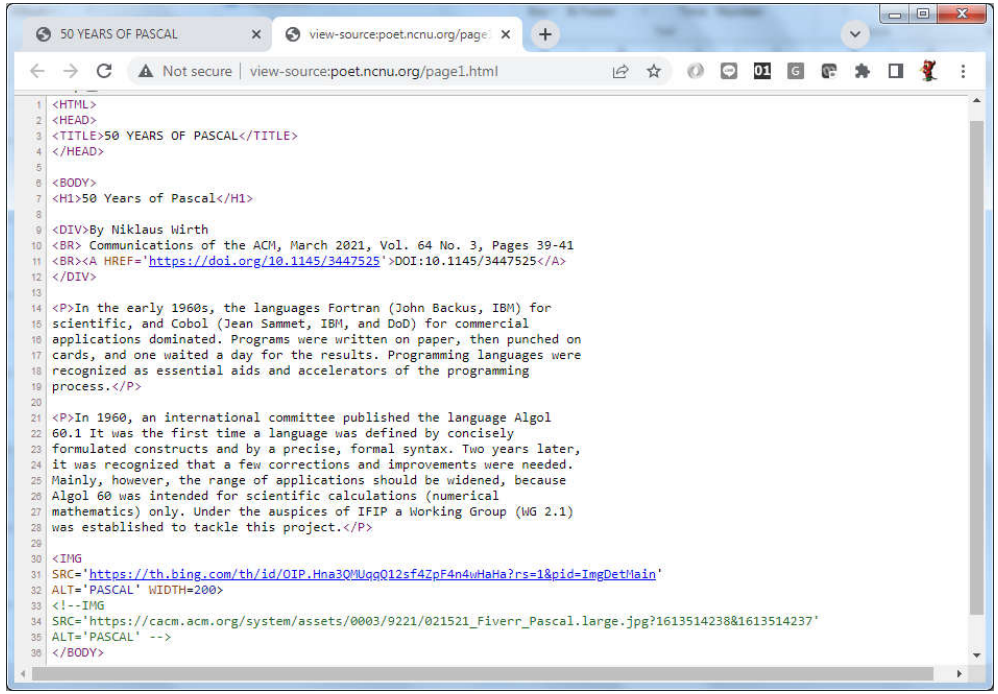
cURL

and the Road to Object-Oriented

Webpages you see in a browser …



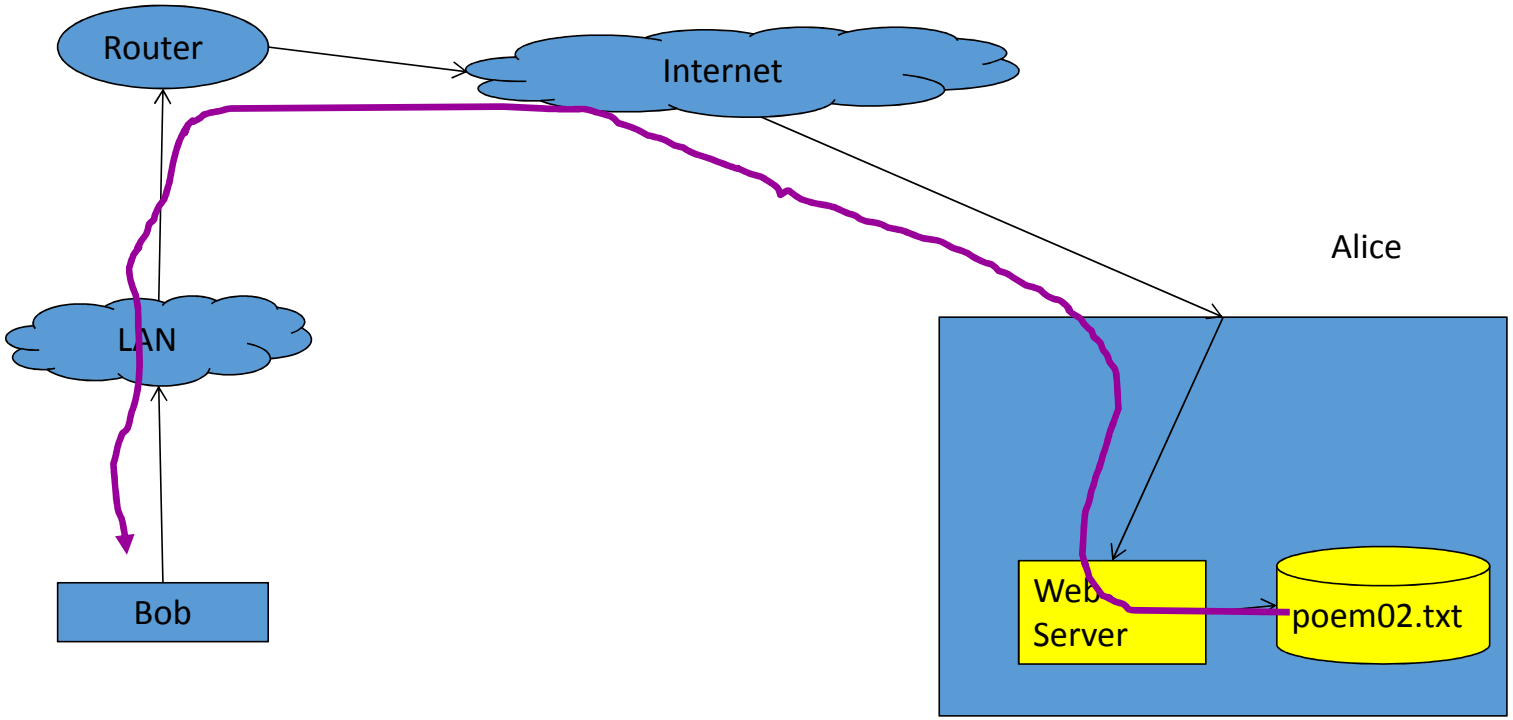
is actually an HTML file interpreted and formatted by software



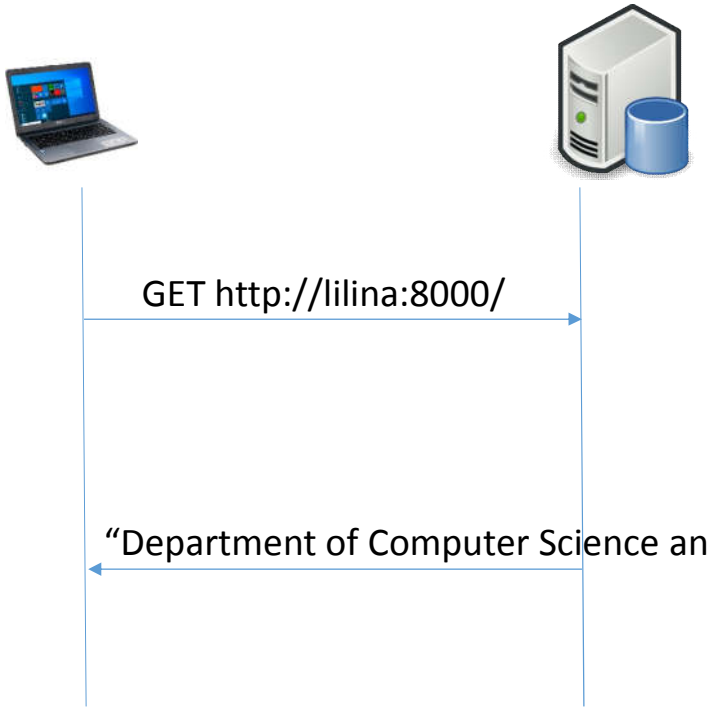
Exercise:

1. Visit a webpage, e.g., http://poet.ncnu.org/page1.html
2. View its HTML source (by Ctrl-U)

Sharing Data on the Internet



Getting Data from a Webpage



You can also do this from the command line.

- curl http://lilina:8000/

```
$ curl http://lilina:8000/
Department of Computer Science and Information Engineering
```

You can do this with a Python program

- from urllib.request import urlopen
- url = 'http://lilina:8000/'
- html = urlopen(url).read().decode()
- print(html)

You can also do this with a C program

```
/* Open a Connection */
#include <stdio.h>
#include <curl/curl.h>
#define URL     "http://lilina:8000/"

int main(void)
{
  CURL *curl;
  CURLcode res;

  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, URL);
    res = curl_easy_perform(curl);
    if(res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));
    curl_easy_cleanup(curl);
  }
  return 0;
}
```

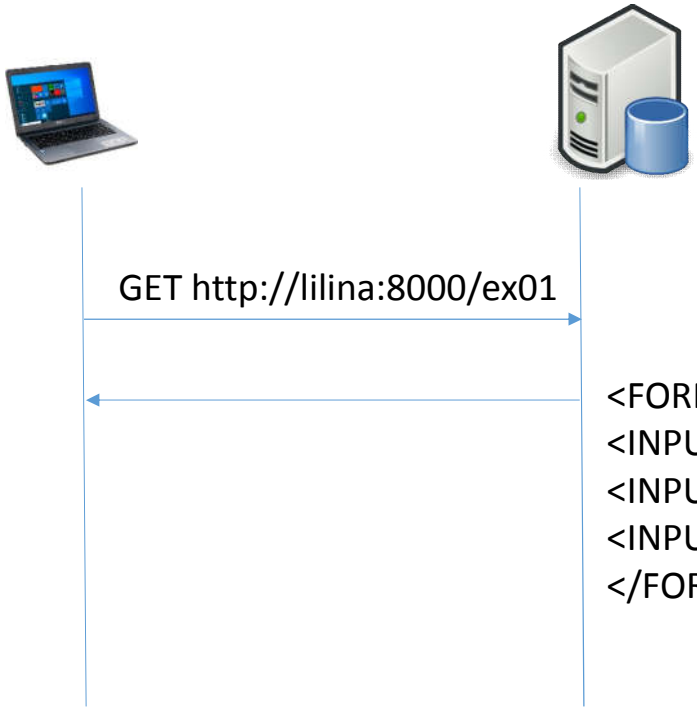
Compile with the option “-lcurl”

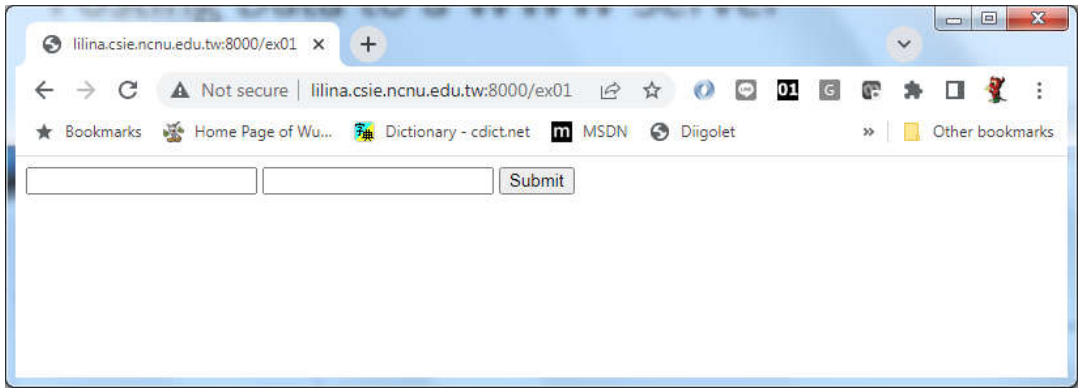
curl-2.c

- The previous program simply retrieves a webpage, and prints out the HTML contents.
- This just mimics the behavior of the “curl” command.

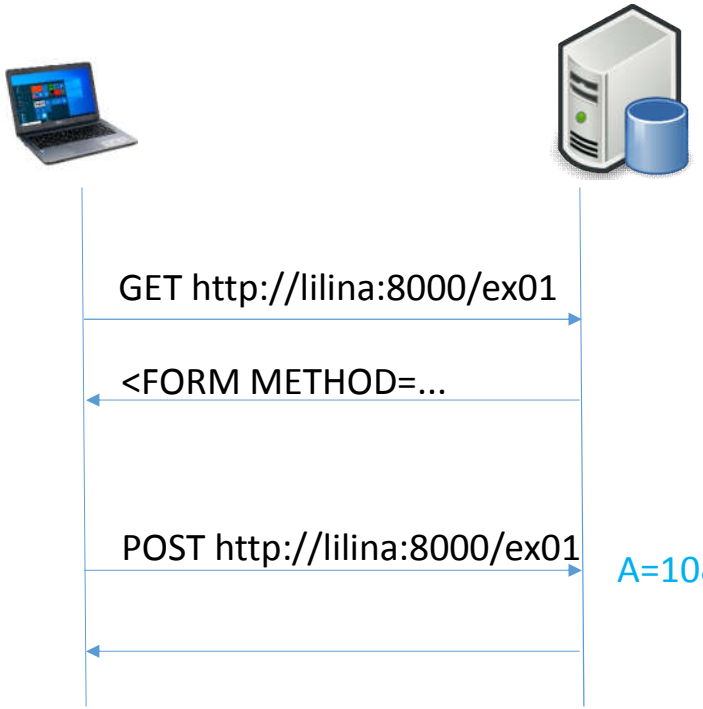
- If you want to store the contents for later usage, you may (dynamically) allocate a memory to store that.
  - See “curl-2.c”

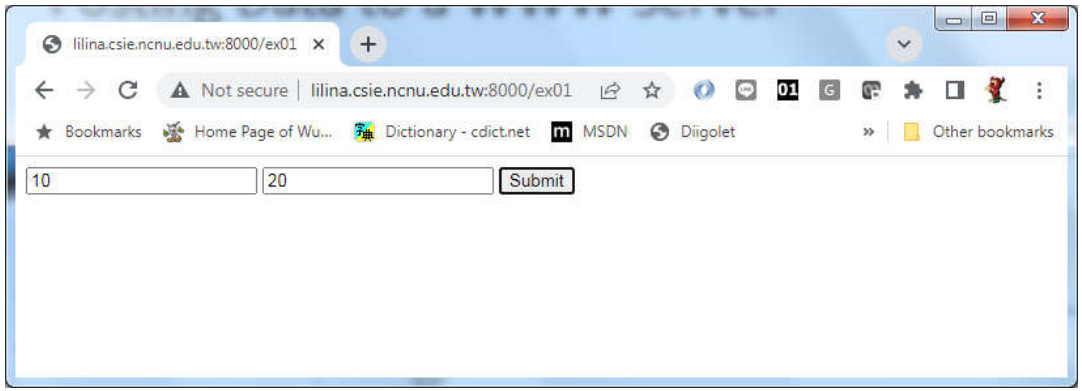
Posting Data to a WWW Server



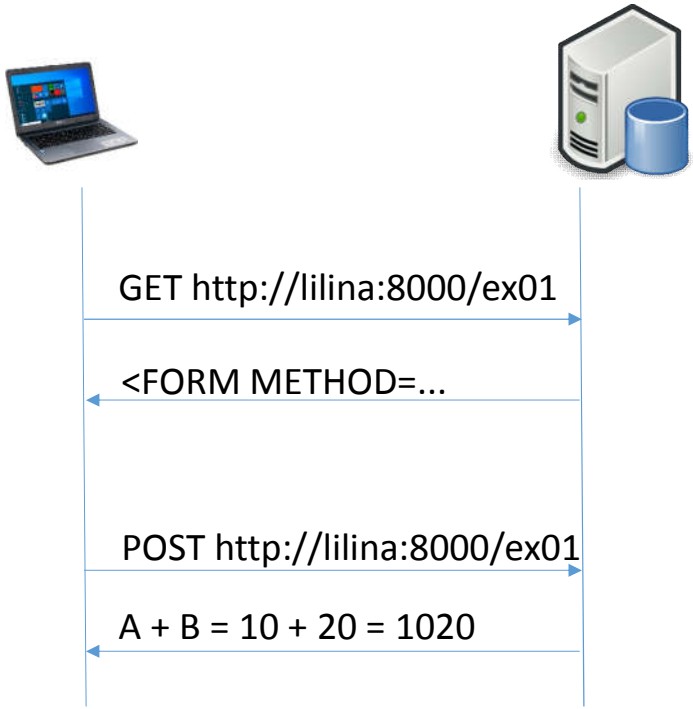


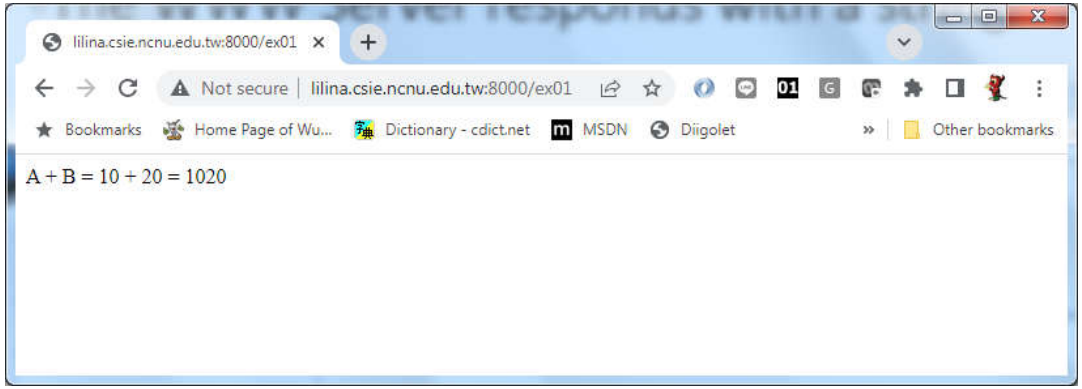
Posting Data to a WWW Server





The WWW Server responds with a string





You can do this from a command line

- curl http://lilina:8000/ex01

```
<FORM METHOD=POST>
<INPUT TYPE=TEXT NAME=A>
<INPUT TYPE=TEXT NAME=B>
<INPUT TYPE=SUBMIT>
</FORM>
```

- curl -X POST -F "A=10" -F "B=20" http://lilina:8000/ex01

```
A + B = 10 + 20 = 1020
```

You can do this from a Python program

```
from urllib.request import urlopen
url = 'http://lilina:8000/ex01'
post_str = 'A=10&B=20'.encode()
html = urlopen(url, post_str).read().decode()
print(html)
```

You can also do this with a C program

```
curl = curl_easy_init();
if(curl) {
  curl_easy_setopt(curl, CURLOPT_URL, WWW);
  curl_easy_setopt(curl, CURLOPT_POSTFIELDS, "A=10&B=20");

  /* Perform the request, res will get the return code */
  res = curl_easy_perform(curl);

  /* Check for errors */
  if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n",
            curl_easy_strerror(res));

  /* Always cleanup */
  curl_easy_cleanup(curl);
}
```

Compile with the option “-lcurl”

curl-4.c

- The previous program simply post data to a WWW server, and prints out the response.
- This just mimics the behavior of the “curl” command.

- If you want to store the response for later usage, you may (dynamically) allocate a memory to store that.
  - See “curl-4.c”

Why do people prefer Python in Webscraping?

- Sir, I know C++ is powerful, but it is just too complicated.
- There are 88 lines of C code in “curl-2.c” to retrieve a webpage.
- With Python, it takes only 4 lines!
- Maybe C program runs faster, but it is faster to write a Python program.

- Not necessarily. That’s why you need to learn Object-Oriented Programming.

Encapsulate curl-4.c as an Object

- class CRequest {
- private:
- string url;
- string response;
- public:
- CRequest& urlopen(string s, string post_str = "");
- string read();
- };

Some details to re-write curl-4.c

- reallocate() returns void*, so it must be reinterpret_cast to char*.
- The main() stores the retrieved webpage in a dynamically allocated memory which is pointed by chunk.memory, so we construct a string with that and return the string as the response.

Now I can use it in my main program

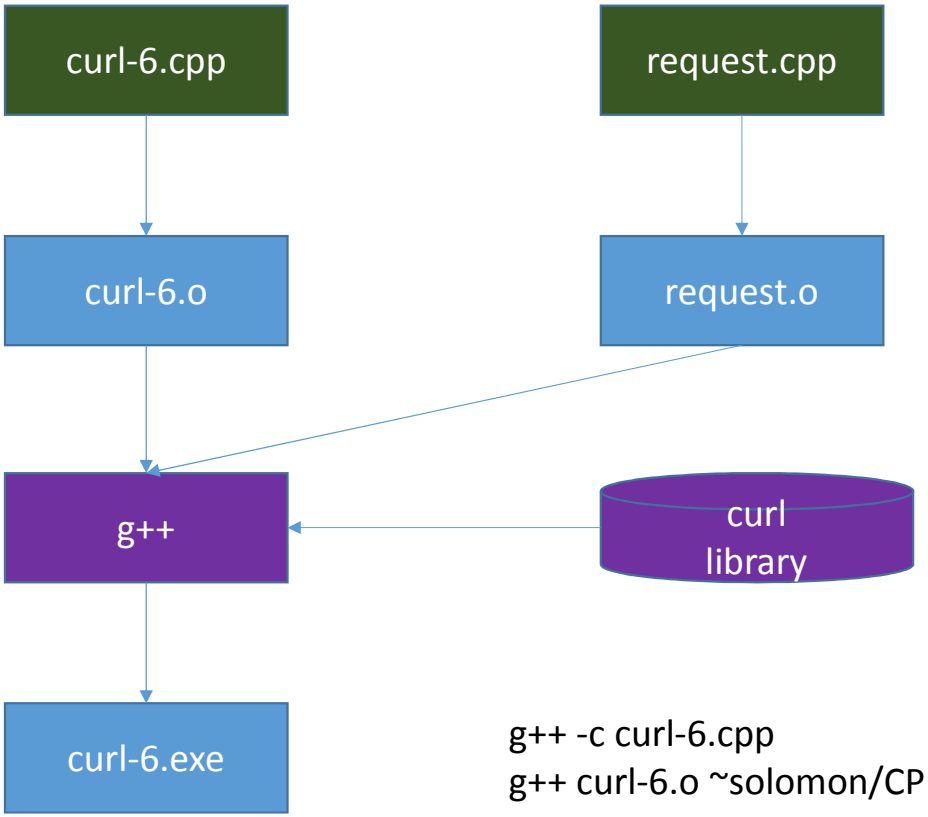
```
int main() {
    CRequest request;
    string html;

    html = request.urlopen(URL).read();
    cout << html << endl;

    html = request.urlopen("http://lilina.csie.ncnu.edu.tw:8000/ex01", "A=5&B=3").read();
    cout << html << endl;
    return 0;
}
```

```
g++ curl-5.cpp -lcurl -o curl-5.exe
```

Separate the Class Implementation

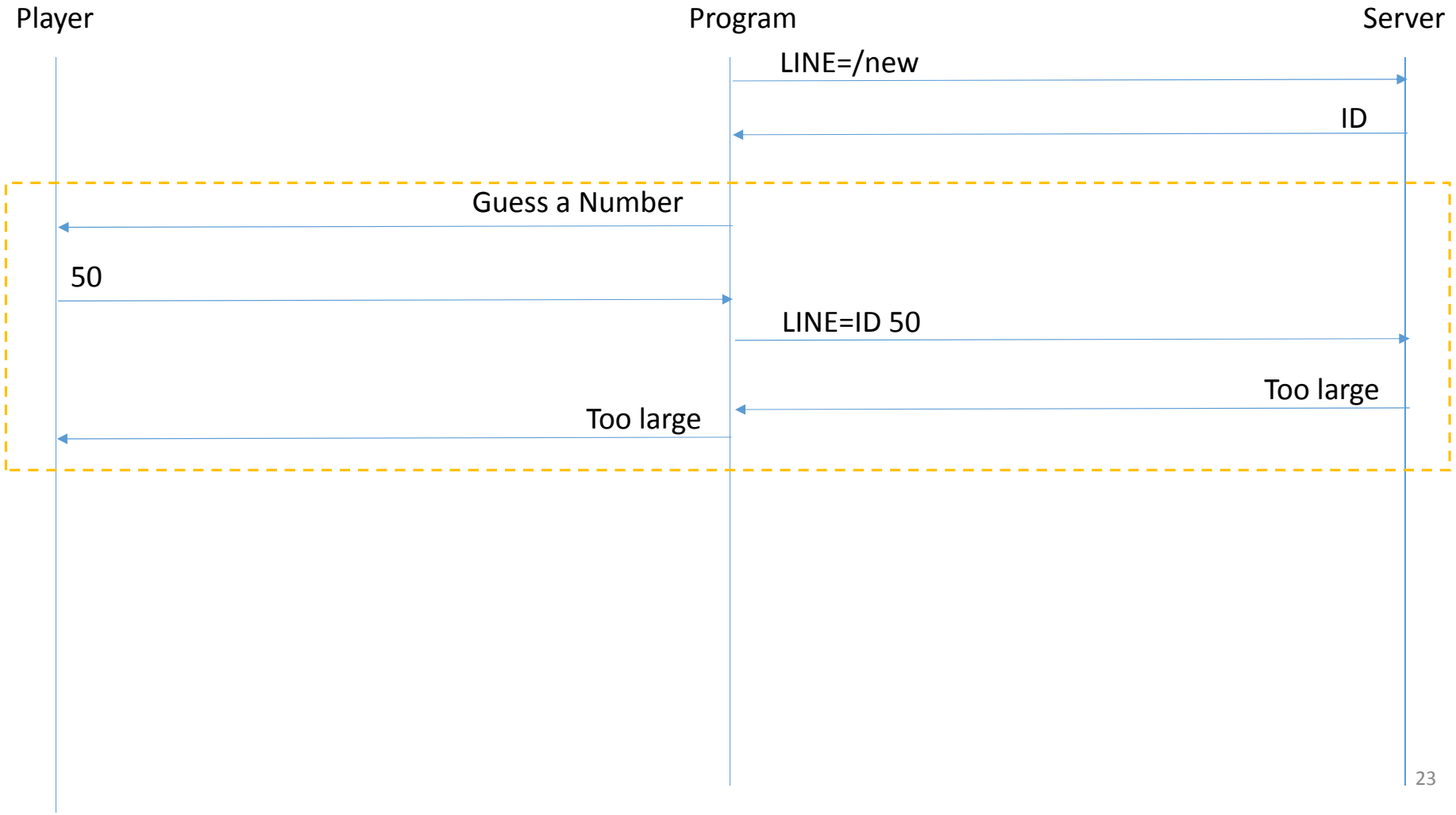


g++ -c curl-6.cpp
g++ curl-6.o ~solomon/CPP/Curl/request.o -lcurl -o curl-6.exe

Hands-On: Login

- Write a C++ program to post a string “STUID=111321xxx” to http://lilina:8000/login.
- You can see the result at http://lilina:8000/list

Exercise: Guess a Number



Exercise: Guess a Number