Lab 2.2

Internet application development using C

Objectives

After completing this lab, you will be able to:

- Apply the basic concepts of Internet application development using C.
- Convey most of a C program into its network-equivalent version.
- Modify advanced C programs available from the Internet and turn them into network applications.

Synopsis

Your ability to develop C Internet applications (CIA) or modify existing C programs into CIAs will greatly enhance your personal and professional profile. In general, Internet applications can reach a far larger audience, including clients, students and friends, than any standalone applications. However, Internet application development is seldom covered in traditional C or network programming courses. In this lab we will show you how to develop your own Internet applications using C. You will find that it is straightforward to convert existing C programs into CIAs (provided that the original C programs are well structured).

Prerequisites

- You have a user account with remote login privileges on the LabBook Support Server, which has been configured to handle CGI scripts under your personal homepage directory.
- Your are familiar with the use of an SSH client such as PuTTY.
- Your PC is connected to the Internet.
- You are familiar with basic UNIX commands.
- You have a basic understanding of C programming and HTML. (Otherwise, consult the online tutorial on C programming mentioned under ‘Background reading and preparation’ below, before doing this lab.)

Background reading and preparation

If you are not familiar with the basics of CGI, consult the online resources listed in Lab 1.8.

If you are not familiar with the basics of C, consult the following online tutorial on C programming:

- ‘C tutorial’ — http://www.cprogramming.com/tutorial.html#ctutorial

  Depending on your knowledge of C, you might need to spend quite a few hours on this tutorial.

Expected duration

Approximately 80 minutes (excluding background reading and preparation)

Procedure

Step 1 Developing your first Internet application in C

Launch PuTTY (or another SSH client that you prefer) and connect to the LabBook Support Server as in Lab 1.1.

In Lab 1.8 Step 1, you created a `public_html` directory under your home directory. This became the homepage directory of your personal website on the LabBook Support Server. Verify the existence of this homepage directory as follows.

```
[s1234567@labsupport ~]$ cd
[s1234567@labsupport ~]$ ls -l
drwxr-xr-x  2 s1234567 ct212-apr09  4096 May  3 20:02 public_html
[s1234567@labsupport ~]$ █
```

There should also be a `cgi-bin` directory under `public_html`. If you have removed these two directories, create them again and set up the proper access permissions for them by following the instructions in Lab 1.8 Step 1.

In the first step of the current lab, you will develop your first C Internet application. The program is relatively simple and does not require any user input. It calculates the first 20 Fibonacci numbers and prints them to the output.[1]

The console version of this program is shown in the listing below. Use your favourite text editor to enter the program and save the file as `fibonacci.c`.

[1] If you don’t know about Fibonacci numbers, read the article http://en.wikipedia.org/wiki/Fibonacci_Number.

```
1   /*
2      File: fibonacci.c
3      Lab 2.2 - Program 1 - Fibonacci Numbers (Console version)
4      This program calculates the first MAX_N Fibonacci numbers and prints them to the
5      standard output (console).
6   */
7   #include <stdio.h>
8
9   #define PROGRAM_NAME "Lab 2.2 - Program 1 - Fibonacci Numbers"
10  #define MAX_N 20
11
12  int main() {
13     int n = 0;
14     int last1 = 0;
15     int last2 = 1;
16     int current;
17
18     printf("%s \n", PROGRAM_NAME);
19     printf("The first %d Fibonacci numbers are:\n", MAX_N);
20
21     while (n < MAX_N) {
22       if (n == 0) {
23         current = 0;
24       } else if (n == 1) {
25         current = 1;
26       } else {
27         current = last2 + last1;
28       }
29       printf("%d, ", current);
30       last1 = last2;
31       last2 = current;
32       n++;
33     }
34
35     printf("...\n");
36     return 0;
37  }
```

Listing 2.2.1 `fibonacci.c`

Note: The line numbers in this lab's program listings are not part of the programs. They are shown for the convenience of explaining the programs.

Compile `fibonacci.c` and save the executable as `fibonacci`. Then run `fibonacci` and observe its output.

```
[s1234567@labsupport cgi-bin]$ gcc -o fibonacci fibonacci.c
[s1234567@labsupport cgi-bin]$ ls -l
total 16
-rwxr-xr-x 1 s1234567 ct212-apr07 4992 May 13 03:17 fibonacci
-rw-r--r-- 1 s1234567 ct212-apr07  672 May 13 03:16 fibonacci.c
-rwxr-xr-x 1 s1234567 ct212-apr07   86 May 11 23:37 hello.cgi

[s1234567@labsupport cgi-bin]$ ./fibonacci
Lab 2.2 - Program 1 - Fibonacci Numbers
The first 20 Fibonacci numbers are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, ...
[s1234567@labsupport cgi-bin]$ █
```

If you want to see more Fibonacci numbers, you may change the constant `MAX_N` defined on line 10. For example, to display the first 30 Fibonacci numbers, you change line 10 to the following:

```
#define MAX_N 30
```

Next, enter the following program into a new file, `fibonacci.cgi.c` in the `cgi-bin` directory. This is the CIA version of the program shown in Listing 2.2.2.

```c
/*
   File: fibonacci.cgi.c
   Lab 2.2 - Program 1 - Fibonacci Numbers (CIA version)
   This program calculates the first MAX_N Fibonacci numbers and prints them to the
   browser output.
*/
#include <stdio.h>

#define PROGRAM_NAME "Lab 2.2 - Program 1 - Fibonacci Numbers"
#define MAX_N 20

void printHeader(void) {
   printf("Content-type: text/html\n\n");
   printf("<html> \n");
   printf("<head> \n");
   printf("<title>%s</title> \n", PROGRAM_NAME);
   printf("</head> \n");
   printf("<body style='padding: 25px;'> \n");
}

void printFooter(void) {
   printf("</body> \n");
   printf("</html> \n");
}

int main() {
   int n = 0;
   int last1 = 0;
   int last2 = 1;
   int current;

   printHeader();
   printf("<h2>%s</h2> \n", PROGRAM_NAME);
   printf("The first %d Fibonacci numbers are: \n", MAX_N);
   printf("<br /> \n");

   while (n < MAX_N) {
     if (n == 0) {
       current = 0;
     } else if (n == 1) {
       current = 1;
     } else {
       current = last2 + last1;
     }
     printf("%d, ", current);
     last1 = last2;
     last2 = current;
     n++;
   }

   printf("...\n");
   printFooter();
   return 0;
}
```

Listing 2.2.2 `fibonacci.cgi.c`

In the above listing, lines that are different from the console version are highlighted. By comparing Listings 2.2.1 and 2.2.2, it should be apparent to you that converting the existing console program into the corresponding CIA version is relatively straightforward. In addition, pay attention to line 13 of Listing 2.2.2:

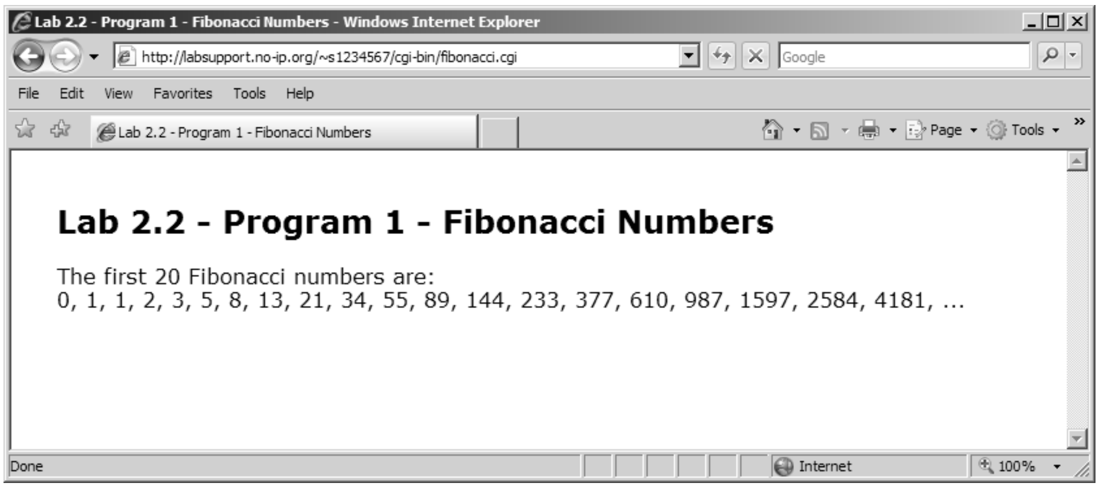
```
printf("Content-type: text/html\n\n");
```

In fact, in any CIA program, the text `"Content-type: text/html\n\n"` *must* be the very first thing to be sent to the browser output. This is a requirement for all CIA programs.

Compile `fibonacci.cgi.c` and save the executable as `fibonacci.cgi`. Then visit the following URL:

http://labsupport.no-ip.org/~s1234567/cgi-bin/fibonacci.cgi

You should see the following output.


Lab 2.2 - Program 1 - Fibonacci Numbers - Windows Internet Explorer
http://labsupport.no-ip.org/~s1234567/cgi-bin/fibonacci.cgi
Google
File Edit View Favorites Tools Help
Lab 2.2 - Program 1 - Fibonacci Numbers
Page Tools
Done
Internet
100%

Lab 2.2 - Program 1 - Fibonacci Numbers

```
The first 20 Fibonacci numbers are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, ...
```

Figure 2.2.1

Congratulations! Your first CIA program is now available to everyone on the Internet!

You have just seen the basic idea behind CIA development. It isn’t difficult and we trust you will agree! When the Web server receives a request to the URL where the CIA program is hosted, the Web server executes the CIA program. The CIA program sends its output back to the Web server, which in turn relays the output to the Web browser that made the original request.

The real power of CIA programming, however, is to facilitate dynamic content — content that varies depending on some predefined conditions and/or user input. In the following steps, you will develop more CIA programs that demonstrate such interactive and dynamic content.

Step 2 Developing a CIA program that accepts user input

In this step, you will modify Program 1 in the previous step to let the user enter the number of Fibonacci numbers to be displayed. In order to do so, we also need an HTML input form to let the user enter the number. This HTML document is relatively simple and is shown in the following listing.

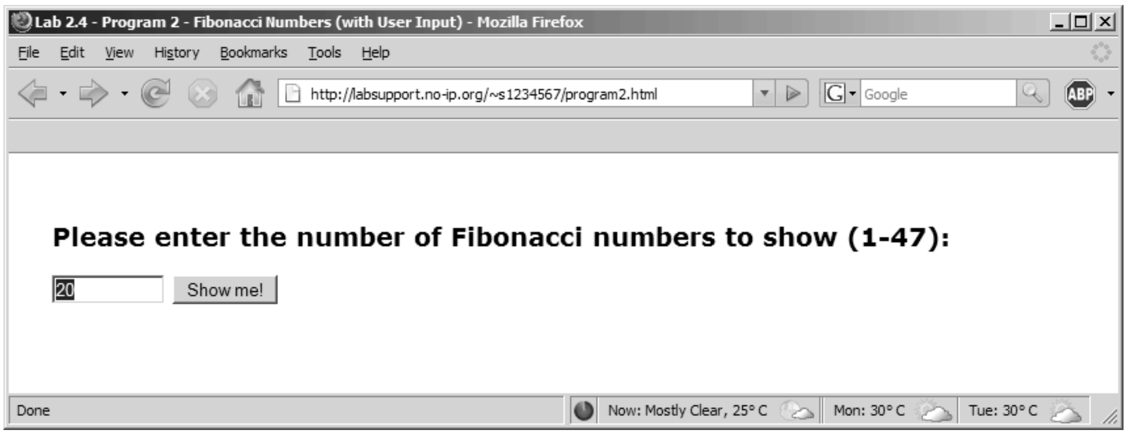
```
<html>
<head>
<title>Lab 2.2 - Program 2 - Fibonacci Numbers (with User Input)</title>
<script language="JavaScript">
function validateInput() {
   var n = parseInt(document.FibonacciForm.max_n.value);
   if (n != NaN && n >= 1 && n <= 47)
     return true;
   else {
     alert("The number must be between 1 and 47 (inclusive).");
     setFocus();
     return false;
   }
}
function setFocus() {
   document.FibonacciForm.max_n.select();
   document.FibonacciForm.max_n.focus();
}
</script>
</head>
<body style="padding: 25px" onload="setFocus()">
<form name="FibonacciForm" method="post" action="cgi-bin/fibonacci2.cgi"
       onsubmit="return validateInput();">
<h3>Please enter the number of Fibonacci numbers to show: (1-47)</h3>
<input name="max_n" value="20" size="10" maxlength="2" />
<input type="submit" value="Show me!" />
</form>
</body>
```

Listing 2.2.3 `program2.html`

Save the above HTML document as `program2.html` under your `public_html` directory. Then access the webpage via the URL:

http://labsupport.no-ip.org/~s1234567/program2.html

The webpage should look like the one below.


Lab 2.4 - Program 2 - Fibonacci Numbers (with User Input) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://labsupport.no-ip.org/~s1234567/program2.html
Google
Show me!
Done
Now: Mostly Clear, 25° C
Mon: 30° C
Tue: 30° C

Please enter the number of Fibonacci numbers to show (1-47):

20

Figure 2.2.2

Next, enter the following program into a new file, `fibonacci2.cgi.c` in the `cgi-bin` directory. It is modified from the CIA version of Program 1. New and modified lines are highlighted.

```
/*
   File: fibonacci2.cgi.c
   Lab 2.2 - Program 2 - Fibonacci Numbers (CIA version)
   This program calculates the first n Fibonacci numbers (where n is submitted via an
   input form) and prints them to the browser output.
*/
#include <stdio.h>

#define PROGRAM_NAME "Lab 2.2 - Program 2 - Fibonacci Numbers (with User Input)"

/* We know we have only one input field "max_n". */
int getInput(void) {
   char input[9];
   fgets(input, 9, stdin);
   return atoi(input + 6);
}

void printHeader(void) {
   printf("Content-type: text/html\n\n");
   printf("<html> \n");
   printf("<head> \n");
   printf("<title>%s</title> \n", PROGRAM_NAME);
   printf("</head> \n");
   printf("<body style='padding: 25px;'> \n");
}

void printFooter(void) {
   printf("</body> \n");
   printf("</html> \n");
}

int main() {
   int n = 0;
   int last1 = 0;
   int last2 = 1;
```

```
36    int current;
37    int max_n = getInput();
38
39    printHeader();
40    printf("<h2>%s</h2> \n", PROGRAM_NAME);
41    printf("The first %d Fibonacci numbers are: \n", max_n);
42    printf("<br /> \n");
43
44    while (n < max_n) {
45      if (n == 0) {
46        current = 0;
47      } else if (n == 1) {
48        current = 1;
49      } else {
50        current = last2 + last1;
51      }
52      printf("%d, ", current);
53      last1 = last2;
54      last2 = current;
55      n++;
56    }
57
58    printf("...\n");
59    printFooter();
60
```

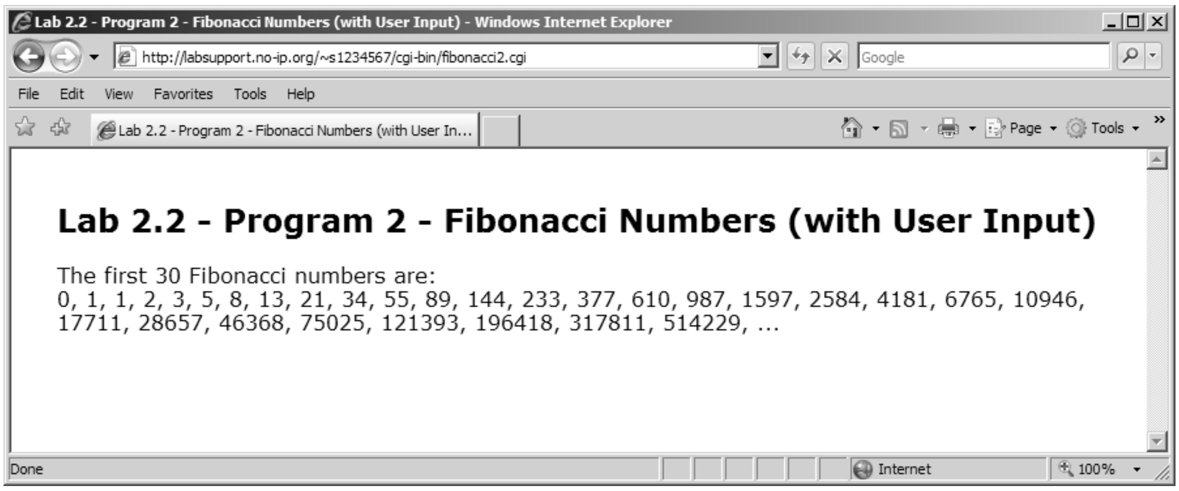
Listing 2.2.4 `fibonacci2.cgi.c`

The function `getInput` on lines 12–16 deserves some elaboration.

```
int getInput(void) {
    char input[9];
    fgets(input, 9, stdin);
    return atoi(input + 6);
}
```

The purpose of `getInput` is to retrieve the number of Fibonacci numbers to be displayed, which is submitted via the input form shown in Listing 2.2.3. In general, for an HTML form which uses the *post* method, the submitted data are passed to the CGI program as a string, `field1=value1&field2=value2&…` via the standard input stream (`stdin`), and the length (in bytes) of the data is passed in an environment variable named `CONTENT_LENGTH`. In our input form, we have only a single input field `max_n` which has a maximum length of two, and it is ensured by the client-side validation JavaScript to be a number in the range of 1 to 47 (inclusive). Thus, the data passed into our CGI program must be a string of the form `"max_n=number"`, where `number` is in the range of 1 to 47 (inclusive). As such, this string contains a maximum of eight characters. Referring back to `getInput`, the line `fgets(input, 9, stdin)` reads the data passed in via `stdin` into the character array `input`. Since the string passed in contains a maximum of eight characters, the character array `input` will contain a maximum of nine characters, including the terminating null character `\0`. The following line `return atoi(input + 6)` converts `input` into an integer, starting from the seventh character until the terminating null character `\0` and then returns the result.

Compile `fibonacci2.cgi.c` and save the executable as `fibonacci2.cgi`. In the input form at http://labsupport.no-ip.org/~s1234567/program2.html, enter a number between 1 and 47 (inclusive) in the text box and then click the **Show me!** button. The output should look like the following.


Lab 2.2 - Program 2 - Fibonacci Numbers (with User Input) - Windows Internet Explorer
http://labsupport.no-ip.org/~s1234567/cgi-bin/fibonacci2.cgi
Google
File Edit View Favorites Tools Help
Lab 2.2 - Program 2 - Fibonacci Numbers (with User In...
Page Tools
Done
Internet
100%

Lab 2.2 - Program 2 - Fibonacci Numbers (with User Input)

```
The first 30 Fibonacci numbers are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946,
17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, ...
```

Figure 2.2.3

Step 3 Developing a CIA program with file manipulation

In this final step, you will modify `fibonacci2.cgi.c` so that the new program will also display a counter telling the number of times the program has been accessed. The counter is implemented using a simple text file which will be created if it does not exist. If the counter file does not contain a valid counter value, the counter value will be treated as 0. Every time the program is accessed, the value contained in the counter file will be incremented by 1.

Copy `fibonacci2.cgi.c` to `fibonacci3.cgi.c` and modify the lines as highlighted in the following program listing.

```
/*
   File: fibonacci3.cgi.c
   Lab 2.2 - Program 3 - Fibonacci Numbers (CIA version)
   This program calculates the first n Fibonacci numbers (where n is submitted via an
   input form) and prints them to the browser output. In addition, it keeps a counter
   of how many times this program has been accessed.
*/
#include <stdio.h>

#define PROGRAM_NAME "Lab 2.2 - Program 3 - Fibonacci Numbers (with User Input and Counter)"
#define COUNTER_FILE "counter.txt"

/* This function increments the counter and returns the new counter value */
   int incrementCounter(void) {
   int counter = 0;

/* Opens the counter file if it exists; otherwise creates the file for read/write */
   FILE* fp = fopen(COUNTER_FILE, "r+");
   if (!fp) {
     fp = fopen(COUNTER_FILE, "w+");
   }

   /* Reads the current counter value and increment it. If the counter file is new
      or the counter value is invalid, the counter value of 0 is used. */
   fscanf(fp, "%d", &counter);
   counter++;

   /* Overwrites the current counter value with the incremented one */
   fseek(fp, 0, SEEK_SET);
   fprintf(fp, "%d", counter);

   /* Remember to close the file before exiting the function! */
   fclose(fp);
   return counter;
}

/* We know we have only one input field "max_n". */
int getInput(void) {
   char input[9];
   fgets(input, 9, stdin);
   return atoi(input + 6);
}

```

```
void printHeader(void) {
   printf("Content-type: text/html\n\n");
   printf("<html> \n");
   printf("<head> \n");
   printf("<title>%s</title> \n", PROGRAM_NAME);
   printf("</head> \n");
   printf("<body style='padding: 25px;'> \n");
}

void printFooter(void) {
   printf("<h4>This program has been accessed ");
   printf("<font color='steelblue'><b>%d</b></font> times.</h4> \n", incrementCounter());
   printf("</body> \n");
   printf("</html> \n");
}

int main() {
   int n = 0;
   int last1 = 0;
   int last2 = 1;
   int current;
   int max_n = getInput();

   printHeader();
   printf("<h2>%s</h2> \n", PROGRAM_NAME);
   printf("The first %d Fibonacci numbers are:\n", max_n);
   printf("<br /> \n");

   while (n < max_n) {
     if (n == 0) {
       current = 0;
     } else if (n == 1) {
       current = 1;
     } else {
       current = last2 + last1;
     }
     printf("%d, ", current);
     last1 = last2;
     last2 = current;
     n++;
   }

   printf("...\n");
   printFooter();
   return 0;
}
```

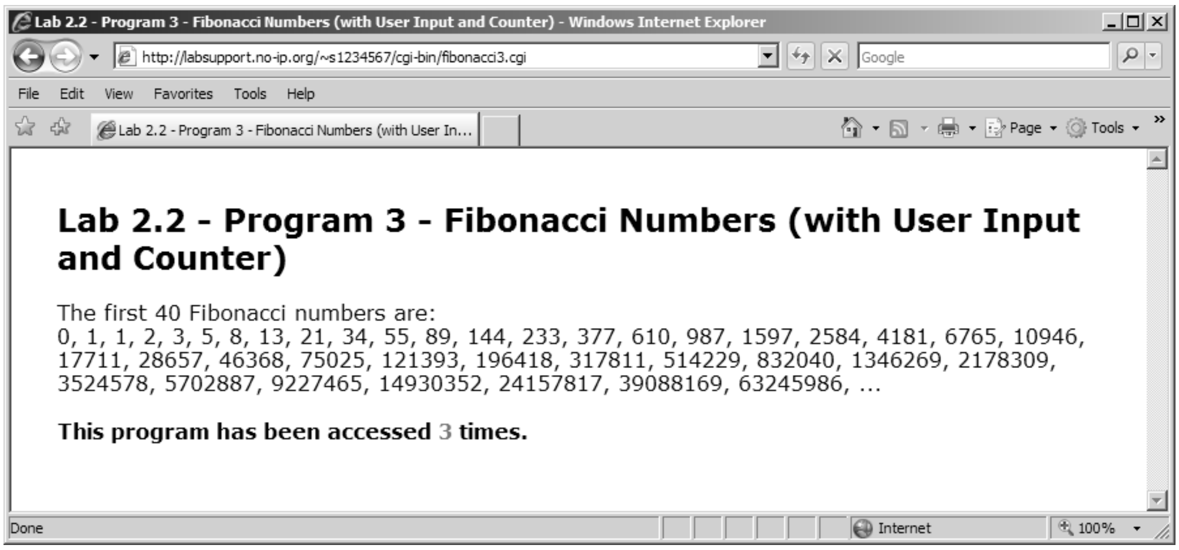
Listing 2.2.5 `fibonacci3.cgi.c`

Compile `fibonacci3.cgi.c` and save the executable as `fibonacci3.cgi`. Then modify `program2.html` as follows and save the modified file as `program3.html`.

```
<html>
<head>
<title>Lab 2.2 - Program 3 - Fibonacci Numbers (with User Input and Counter)</title>
<script language="JavaScript">
function validateInput() {
   var n = parseInt(document.FibonacciForm.max_n.value);
   if (n != NaN && n >= 1 && n <= 47)
     return true;
   else {
     alert("The number must be between 1 and 47 (inclusive).");
     setFocus();
     return false;
   }
}
function setFocus() {
   document.FibonacciForm.max_n.select();
   document.FibonacciForm.max_n.focus();
}
</script>
</head>
<body style="padding: 25px" onload="setFocus()">
<form name="FibonacciForm" method="post" action="cgi-bin/fibonacci3.cgi"
       onsubmit="return validateInput();">
<h3>Please enter the number of Fibonacci numbers to show (1-47):</h3>
<input name="max_n" value="20" size="10" maxlength="2" />
<input type="submit" value="Show me!" />
</form>
</body>
```

Listing 2.2.6 `program3.html`

Open http://labsupport.no-ip.org/~s1234567/program3.html in your browser and test the new program. This time the output should also display how many times the program has been accessed.


Lab 2.2 - Program 3 - Fibonacci Numbers (with User Input and Counter) - Windows Internet Explorer
http://labsupport.no-ip.org/~s1234567/cgi-bin/fibonacci3.cgi
File Edit View Favorites Tools Help
Done
Internet
100%

Lab 2.2 - Program 3 - Fibonacci Numbers (with User Input and Counter)

```
The first 40 Fibonacci numbers are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946,
17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309,
3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986, ...

This program has been accessed 3 times.
```

Figure 2.2.4

Hit the **Reload** button of your browser a few times. Make sure the counter is incremented by 1 after each reload. In addition, try removing the counter file or changing the content of the counter file. Observe the resulting output of the program after the respective changes.

You have successfully finished this lab.

Summary

In this lab, you learned how to modify a C console program into an Internet application that is accessible by all users over the Internet. You also learned how to pass data submitted via an HTML input form into your C Internet application program. You should have found that it is straightforward to convert an existing C program into its CIA equivalent version, provided that the original C program is well-structured. The programs that you developed in this lab are relatively simple ones, but we trust that they can illustrate the important concepts of CIA development.

Further reading and useful resources

- ‘Getting started with CGI programming in C’ — http://www.cs.tut.fi/~jkorpela/forms/cgic.html

Questions and exercises

1 Search the Internet to find out what programming languages are commonly used to implement CGI programs.

2 Search the Internet to find out what other technologies are commonly used to provide dynamic Web content.

3 On line 13 of `fibonacci2.cgi.c`, why is the character array `input` defined as having nine elements?

```
char input[9];
```

Can we use a smaller or larger array to hold the data passed in?

4 On line 15 of `fibonacci2.cgi.c`, we use the function `atoi` to retrieve the number we need:

```
return atoi(input + 6);
```

What does `input + 6` mean?

5 In `fibonacci2.cgi.c` and `fibonacci3.cgi.c`, `getInput` is hard-coded to handle a single input field `max_n` with a maximum length of 2. Modify `getInput` so that it can handle multiple input fields with varying lengths. It will take a single argument specifying the name of the field to be retrieved. The value of the specified field will be returned as a string.

6 a Write a program in C (or C++, shell script, Perl or Python) to count the frequency of a certain key pattern in a given database file. The database file is a plain text file which contains a 7 by 3 matrix of numbers from 1 to 49.

```
2 1 37
5 3 24
1 3 36
1 2 13
3 5 26
9 2 24
8 49 9
```

The program takes three numeric keys, which are separated by spaces, entered from the console (`stdin`). The program then prints the number of occurrences of each of the input keys in the database file.

Here is an example run of the finished program.

```
[s1234567@labsupport ~]$ ./q6a
Please enter 3 keys (separated by space): 7 1 24
The key '7' is found 0 times in the database
The key '1' is found 3 times in the database
The key '24' is found 2 times in the database

[s1234567@labsupport ~]$ █
```

b Convert the frequency count program to a CIA program so that it can be accessed via the Web.