Design and Implementation of Embedded Web Server Based on ARM and Linux

Yakun Liu Xiaodong Cheng
College of Electronic Information Engineering
Inner Mongolia University
Hohhot, P.R. China
yakun_liu@yahoo.com.cn cxd0808@imu.edu.cn

***Abstract*—This paper achieves the design of an embedded Web server, which takes ARM920T-S3c2410s chip as its core and Linux as its operating system. This is because Linux can be reduced and transplanted. The method used to transplant Web server Boa on the embedded Linux platform is also discussed in detail, and through CGI technology functions of dynamic Web page is successfully realized. Relevant experiments show that after the Web server is embedded into the network video monitoring system, dynamic page interaction can be achieved between the Web server and the embedded system via the browser in the Windows environment.**

Keywords-embedded systems; linux; embedded Web server; Boa; CGI

I. INTRODUCTION

With the rapid development of Internet information technology, those fieldbus and Industrial Ethernet which are of high-specialization and high cost and are used in control areas are gradually being replaced by Ethernet [1]. Embedded systems and Internet technology are combined to form a new technology – the Embedded Internet Technology, which developed with the popularization of computer network technology in recent years [2]. Without restrictions from devices and systems, this technology could function in the hardware and software as long as they are connected. Only by using web browser through the Ethernet and TCP/IP protocol can users get access to various information [3]. It brings great convenience to remote video monitoring and equipment management. The main advantages of using embedded Web server mainly include: (1) the client can be freely set and the browser can be used directly without installing additional client software; (2) for the harmonization of Web standards, it is possible to develop cross-platform transplantation; (3) the operating system Linux, which can be reduced and transplanted, provides a convenient, fast and simple method for embedded systems and Internet access [4].

II. OVERVIEW OF EXPERIMENTAL PLATFORM FOR THE EMBEDDED WEB SERVER

The design in this paper applies S3C2410s32-bit ARM microprocessor which takes ARM920T as its core. This microprocessor has rich resources, including Clock, USB, SDRAM, UART, Nand Flash, LCD, RS232 Interface, Ethernet Interface, JTAG, Power, etc. These modules can help achieve Internet services. The logical structure of the hardware is shown in Fig. 1.

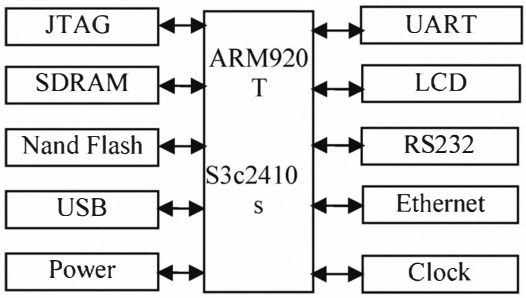

Figure 1. The structure of system hardware

III. EMBEDDED WEB SERVER

A. The system diagrame of Embedded Web server

The system structure of embedded Web server is shown in Fig. 2. The entire system uses B/S mode. The client PC is connected to the Internet through a browser and then gets access to the embedded Web server. Through this way, remote login and operation are realized [5]. Compared with the traditional C/S mode, this mode is simple to use, convenient to maintain, and easy to extend.

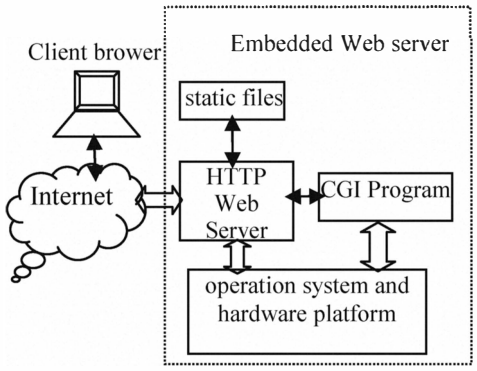

Figure 2. The system diagram of Embedded Web server

B. The choice of Embedded Web server

Generally speaking, the embedded devices have limited resources and don't need to handle the requests of many users simultaneously. Therefore they do not need to use the most commonly used Linux server Apache. Web server which is specifically designed for embedded devices are applied in such case [6]. This kind of Web server requires relatively small storage space and less memory to run, which makes it quite suitable for embedded applications.

The typical embedded Web server has three kinds, namely httpd, Boa and thttpd [7]. As the simplest Web server, httpd has the weakest functions among the three. It does not support authentication and CGI technology while

Boa and thttpd support these functions [8]. If Web server only provides some static web pages such as simple on-line help and system introduction, then a static server can be adopted; if you need to improve system security or interact with users such as real-time status query and landing, then you have to use dynamic Web technologies. In such situation, either Boa or thttpd can achieve these goals. In the present research, we adopt Boa, the Web server suitable for embedded system, because thttpd has less function and needs far more resources to run.

C. *The principle of Embedded Web server Boa*

Boa is a single task Web server. The difference between Boa and traditional Web server is that when a connection request arrives, Boa does not create a separate process for each connection, nor handle multiple connections by copying itself. Instead, Boa handles multiple connections by establishing a list of HTTP requests, but it only forks new process for CGI program. In this way, the system resources are saved to the largest extent [9].

Like a common Web sever, an embedded web server can accomplish tasks such as receiving requests from the client, analyzing requests, responding to those requests, and finally returning results to the client. The following is its work process.

- Complete the initialization of the Web server, such as creating an environment variable, creating socket, binding a port, listening to a port, entering the loop, and waiting for connection requests form a client.
- When there is a connection request from a client, Web server is responsible for receiving the request and saving related information.
- After receiving the connection request, Boa analyzes the request, calls analysis module, and works out solutions, URL target, and information of the list. At the same time, it processes the request accordingly.
- After the corresponding treatment is finished, the Web server sends responses to the client browser and then closes the TCP connection with the client. For different request methods, the embedded Web server Boa makes different responses. If the request method is HEAD, the response header will be sent to the browser; If the request method is GET, in addition to sending the response header, it will also read out from the server the URL target file of the client request and send it to the client browser; If the request method is POST, the information of the list will be sent to corresponding CGI program, and then take the information as a CGI parameter to execute CGI program. Finally, the results will be sent to client browser. Boa's flowchart is shown in Fig. 3.

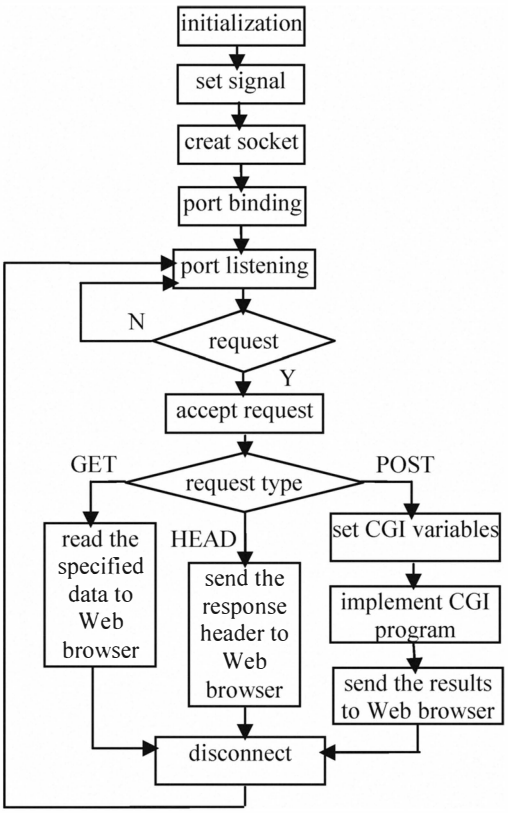

Figure 3. Embedded Web server flowchart

D. *The creation of an embedded Web server*

In the embedded Linux system, the creation of a Web server Boa has the following steps:

1) Download the source code of Boa. The source code can be download from http://www.boa.org [10].

2) Transplant the procedure of Boa. Decompress the downloaded source code and lead it to enter "scr" subdirectory of the source directory [11]*:*

#tar xzvf boa.tar.gz
#cd boa/src
Creat "Makefile" file:
#/configure

Modify "Makefile" file. Mainly modify the cross-compiler, find CC=gcc, change it into CC=armv4l-unknown-linux-gcc, save these changes and quit "Makefile" file.

Specify the root directory path of Web server: enter"boa/scr/" directory, and specify the absolute path of root directory of the Web server by modifying the statements which are in "defines.h" file.
#define SERVER_ROOT"/mnt/yaffs/share/www/boa/http"

Then run "make" to compile, it will creat a file named "boa" in the directory of "boa/src". This file shall be the executable file of Web server Boa.

3) Configure Boa so that it can support the implementation of CGI programs.

Boa requires to establish a boa directory in the root file system"/mnt/yaffs/share/www". A configuration file"boa.conf" will be loaded when the boa boots. This file must be edited before the boa program is running. There is already a sample boa.conf in the Boa source directory. It

can also be modified on its basis. The following configurations need to be changed:

Port 80 //set the port of Web

Group 0 //opening up the restrictions on the user group

……

ErrorLog/mnt/yaffs/share/www/boa/log/boa/error_log //set the actual path of the error log

DocumentRoot/mnt/yaffs/share/www/html //set the home directory of the HTML file

ScriptAlias/cgi-bin/ /mnt/yaffs/share/www/cgi-bin //specify the actual path of the virtual path of the CGI script

ScriptAlias/index.html//mnt/yaffs/share/www/html/index.html //specify the actual path of the virtual path of the server's default page

4) Test whether Boa can work normally, and whether the static HTML pages can be visited normally. In this paper, NFS approach is used to test.

According to the configuration of boa.conf, we copy the tested home page index.html into "/mnt/yaffs/share/www" directory. The IP address of the board is set to be 192.168.0.115. We enter"/mnt/yaffs/share/www/boa/src" through minicom, and then run "./boa", and visit the following website: http://192.168.0.115 on PC browser. Then we could see the pages of "/mnt/yaffs/share/www/index.html". That means Boa works normally in the embedded target systems.

IV. THE IMPLEMENTATION OF DYNAMIC WEB PAGES UNDER LINUX

There are many kinds of technologies such as CGI, ASP, PHP JSP and so on, which are used to achieve dynamic Web pages. If the dynamic pages are to be realized under Linux operating system, CGI is preferred. CGI [12](Common Gateway Interface) is a common interface standard which is applied to interact between the application of external expansion application and Web Server. CGI provides the Web server with a channel to implement external program. This service technology makes browser and server interactive. CGI is the program consistent with this common interface standards and running on the Web server. CGI programs can be produced in any programming language, for example, Shell scripting language, Perl, Fortran, Pascal and C language and so on. The C language is chosen to write CGI programs in the present paper.

The flow chart is shown in Fig. 4, where the client interacts with the on-site I/O module through CGI program.

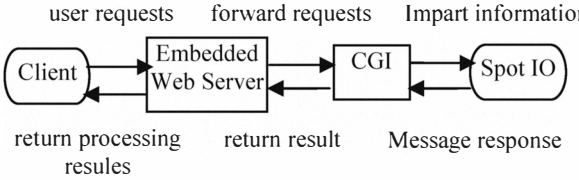

Figure 4. Interaction between the client and the on-site I/O module

After the daemon of Web server receives client requests, a child process will be created. Then this child process will set relevant data requested by CGI as environment variables and meanwhile build two data channels between external CGI program and the server (standard input/output). Then the CGI program assigned by URL is started and keeps pace with the child process in order to monitor the implementation state of CGI program. The result of disposition is passed to the daemon of Web server through the standard output stream by the child process. Then the processing results are reported back to the client by daemon as a response message.

A CGI program is usually divided into two parts. (1) Receive data from submission form according to POST method or GET method. (2) Generate the HTML source code by means of printf() function and then correctly return the decoded data to the browser.

V. EXPERIMENT RESULTS

We load each driver and start the Web server on an already established experimental platform-S3c2410s. In client browser, we input the corresponding IP address http://192.168.0.115/, and then the Web page is opened which is as shown in Fig. 5.

Figure 5. The page when the client Web browser accesses to the Web server

VI. CONCLUSION

This embedded Web server is a separate module which can provide a standard interface. With slight modifications it can be applied easily to embedded fields such as on-site AC servo system, industrial control, and intelligent appliances. Therefore, it has a wide range of application prospects and great promotion value.

The embedded Web server designed in this paper is based on the ARM-Linux operating system. It succeeds in network video monitoring. The whole system has low-cost, good openness and portability, and is easy to maintain and upgrade. The Web server Boa selected in the present research requires small storage space and occupies less memory when it's running. It also has more functions and supports CGI. Communication between external expansion applications and Web server can be achieved through CGI technology. This method can not only improve system security, but also make it possible to interact with users and create dynamic Web pages.

REFERENCES

[1] Wang Xianchun, Guo Jierong, Hu Weiwen, and Fan Xiping, "Design and Implementation of Embedded Web Server Based on ARM and Linux," Micro Computer Information, vol. 23(5-2), 2007, pp. 164-165.

[2] Jacek W, "Embedded Internet technology in process control devices," IEEE Internet Computing, Vol. 34, 2000.

[3] I. Douglas, "Engineering Web Technologies for Embedded Applications," IEEE Internet Computing, May/June 1998.

[4] Wang Tianmiao, Publication: Embedded System Design and Case Development. Beijing: Tsinghua University Press, 2002.

[5] Liu Yingshui, Xiao Zhengyu, and Sun Wei, "Embedded Web Server Based on ARM and Linux," Microcontrollers & Embedded Systems, june 2007, pp. 14-21.

[6] Mi-Joung Choi, Hong-Taek Ju, Hyun-Jun Cha, Sook-Hyang Kim, and J. Won-Ki Hong, "An Efficient Embedded Web Server for Web-based Network Element Management," International Journal of Network Management, Vol. 10, May 2000.

[7] Zhang Wenya., "Design and Implementation of Network Vidio Monitoring System Based on Embedded Linux," Southwest Jiaotong University Master Degree Thesis, May 2009, pp. 32-33.

[8] Li Weixuan, Zhao Jing, Peng Zhicheng, and Xu Zhihao, "Research of Remote Monitoring System Based on Embedded Web Server," AECC Symposium, vol. 20, 2007, pp. 147-148.

[9] Liu Minying, "Design and Realization of a Remote Monitoring System Based on Web", Dalian Maritime University Master Degree Thesis, May 2008. pp. 40-49.

[10] Zhao Huijuan, "The building of embedded Linux development platform based on ARM9 and realization of Boa," Southwest Jiaotong University Master Degree Thesis, Match 2008, pp. 57-59.

[11] R. Stones, and N. Matthew. Publication: Beginning Linux Programming(2nd Edition). Endland: Wrox Press, 2003.

[12] J. Dwight, M. Erwin, R. Niles, CGI Development Handbook. Machinery Industry Press.