采用嵌入式及 SSL 的多用户安全型串口服务器设计①

沈 雁，戴瑜兴，汤 睿

(湖南大学 电气与信息工程学院，长沙 410082)

摘 要：针对目前串口服务器在安全性及功能性上的不足之处，提出以 ARM9 和嵌入式 Linux 为平台，采用 SSL 协议开发的具有多用户管理的安全型串口服务器。介绍了多用户安全型串口服务器的整体硬软件设计方案。实现了 U-boot、内核、文件系统以及各功能库在内的嵌入式 Linux 系统在 AT91SAM9261 处理器上的移植，对软件系统的构架进行了论述，着重介绍了数据处理框架、用户管理以及 SSL 的实现。应用结果表明，此设备运行稳定可靠，能够满足安全通信的要求。

关键词：Linux；ARM9；多用户；串口服务器；SSL

Design of Multi-User and Secure Serial Device Server Based on Embedded System and SSL

SHEN Yan, DAI Yu-Xing, TANG Rui

(College of Electrical and Information Engineering, Hunan University, Changsha 410082, China)

Abstract: For the shortage of current serial device server which almost lacks security and functionality, based on arm9 and Linux, we have designed a secure serial device server using SSL protocol and multi-user management. The overall design of hardware and software were introduced. Including U-boot, kernel, root file system and a variety of tools, embedded Linux system has been emigrated, we discussed the framework of the software system, focusing on data process framework, user management and the implementation of SSL. The application shows that this device runs stably and meets the needs of secure communication.

Key words: Linux; ARM9; multi-user; serial device server; SSL

物联网的出现及迅猛发展为世界经济带来了新的增长点，将那些传统的仪器设备采用统一的方式接入 LAN/Internet 势在必行[1]。而众多的设备采用各种串口方式进行通信，当前出现了称为串口服务器的工具来将这些设备接入 LAN/Internet，一般都采用微控制器实现，其功能比较简单。同时由于互联网在安全性方面存在众多的问题，有许多仪器设备的数据关系到企业乃至国家的信息安全，因此设计与开发具备良好安全性的串口服务器极为重要，本文采用 ARM9 微处理器和 Linux 操作系统，设计开发了具有多用户访问，使用 SSL 协议的安全型串口服务器。

1 硬件电路组成

所采用的主控微处理器为 AT91SAM9261[2],它扩展了 DSP 指令集和 Janelle java 加速器，主时钟频率达到 190MHz，为大量复杂的运算提供了强大了支持。同时，其众多的外部接口包括了从 UART 到 USB 的广泛支持。其自身只带有四个 UART 接口，可以直接用来做为四串口服务器使用。另外，由软件控制的功率管理器（PMC）可以有选择地关闭和开启处理器核心和各种 外设或降低其工作频率，能使得系统的功耗保持在最小。

使用 DM9000A 作为以太网控制芯片，其集成 10/100M 物理层接口；内部带有 16K 字节 SRAM 用作接收发送的 FIFO 缓存；支持 8/16bit 两种主机工作模式；支持 TCP/IP 加速减轻 CPU 负担，提高整机效能。另外做为系统必需，还包括了电源模块、32M 的 SDRAM、2G 的 NAND Flash 以及 4M 的 Data Flash（用

① 收稿时间:2010-10-27;收到修改稿时间:2010-11-20

于存放 U-boot 和 Linux 内核）。系统的硬件框图如图 1 所示。

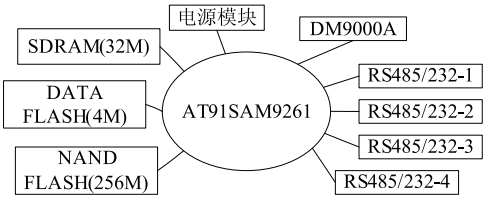

图 1　硬件组成

2　软件平台构建

系统的软件平台基于嵌入式 Linux 操作系统，主要的工作包括引导程序的移植，嵌入式 Linux 内核的编译，及根文件系统的生成。在做这些工作之前，需要建立一个完整的交叉编译环境，包含 gcc、glibc 等等，用来在 PC 机上产生 ARM 处理器能执行的二进制代码。

引导程序采用广泛使用的 U-boot，其功能强大，包含了对 TFTP、NFS 以及 NAND 的支持。同时 ATMEL 发布了专门针对其产生的处理器的 U-boot 修改版本，对快速的产品开发起到了很大的帮助。从其专门网站下载 U-boot 源代码，解压生成 uboot 源代码目录，在其中运行“make at91sam9261ek_config; make”即可生成 uboot.bin 的二进制文件，通过使用 ATMEL 的 ISP 工具将其写入到 DATA FLASH 的 8000 位置，复位 AT91SAM9261 之后即能看到“U-boot>”字样的提示符，表示 U-boot 运行成功。

第二步为编译 Linux 的内核，同样有针对 AT91 SAM926X 型号处理器的专门修改版本，考虑系统的稳定性，采用 Linux-2.6.24 的内核版本。下载之后解压生成 Linux-2.6.24 的目录，运行“make ARCH=arm CROSS_COMPILE=arm-linux-menuconfig”命令，表示使用 Linux 的处理器架构为 ARM，交叉编译器的前端为“arm-linux-”，同时使用菜单的方式对 Linux 进行配置，只需将其中的芯片选为 AT91SAM9261 即可。此后即可用 make 命令编译生成二进制映像文件，为了让 U-boot 能向内核传递一些保正正确启动的参数，二进制映像文件还需要使用 U-boot 的 mkImage 工具进行包裹之后才能被 U-boot 使用。

下一步，还需要制作根文件系统[3]，即以”/”为起点的文件系统，在 PC 机上制作 rootfs 目录来模拟，在其下生成 etc、bin、lib 等 Linux 所需的目录结构，使用 busybox 工具在 bin 目录中生成基本命令工具集，同时将交叉编译器所使用的 glibc 的库文件拷贝到 lib 目录下，再经过对 etc 目录的配置之后，完整的嵌入式 Linux 系统即构建完成。

最后，为了满足应用软件的需要，还需要编译出 openSSL、XML 等支持库，将它们拷贝到根文件系统的/lib 或/usr/lib 目录下，将来应用软件运行时就会自动连接到这些库调用所需要的功能。

3　软件实现

3.1 总体架构

串口服务器的应用软件采用 C 语言进行线程化及模块化设计，总体架构如图 2 所示。

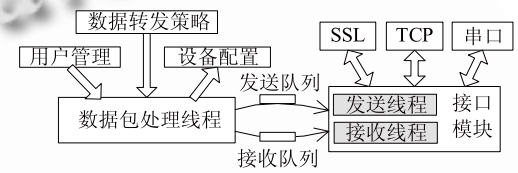

图 2　软件框架

在设计中借用了 Unix/Linux 的设计思想，Unix/Linux 在系统层将所有的设备及接口都视为文件，采用统一的接口规范，使得应用程序只需使用 read、write 等操作即可与设备进行交互，极大地方便了程序的开发及维护，同时具有极高的可扩展性。在本设计中，将串行接口以及 TCP 通信采用统一的接口框架，可以带来扩展支持更多接口的好处，比如 MODBUS 总线协议，CAN 总线等等，另外 SSL 也就是在这个基础上的扩展支持。定义结构体如下：

```
struct Interface{
    int type;
    int fd;            //file descriptor
    int init(int num);
    int (*read)(int clientfd, BufPtr buf, int size);
    int (*write)(int clientfd, BufPtr buf, int size);
    int (*accept)(int sockfd);
    int (*disconn)(int fd);
    ….
}
```

结构体中使用了 Buf 缓冲块结构体，用来存储接收或者发送的数据。同时定义了接口的回调函数比如读写，连接等操作，这些函数根据不同的接口就有不

同的实现。

接口模块与处理模块之间采用队列的方式进行通信，为了充分利用 Linux 系统的特性，采用信号量的方式来模拟出生产者－消费者模型，当队列为空时使得接收线程进入睡眠状态，降低系统资源消耗。数据包处理线程负责根据转发策略来处理来自串口或者以太网的数据、用户的登录以及设备配置操作等等。

3.2 多用户支持及工作模式

为了使得多个用户同时在不同地点都能访问到服务器，设计了用户管理模块，将用户分为两类：一类为管理员用户，除了对串口的通信操作之外，还可以对串口服务器进行配置操作(以太网参数，串口参数等)；另一类为普通用户，它只能与串口进行数据的通信。进行这样的划分之后，可以避免掉许多普通访问者对设备进行篡改，造成不必要的故障。

每个创建的用户都被保存在 user.xml 文件当中，采用的格式如下，其中的 password 节点不以明文的方式来存取，而是使用 MD5 算法对“用户名+密码”字符串生成 16 字节的摘要字串进行保存，这样可以方式被非法获取到密码。

```
<config>
  <user>
    <name>test</name>
    <password>16 字节 MD5</  password>
    <type>normal</type>
  </user>
</config>
```

用户对串口的操作应是互不影响的。用户 A 对串口 1 进行操作的时候，意味得用户 B 此时不能对串口 1 进行操作，但可能可以对串口 2 进行操作，如果串口 2 处在空闲的状态下。因此，对串口操作定义两种工作模式：

（1）响应模式，在此模式下，定义用户对串口（也即串口所接设备）发送请求数据和串口对用户的请求回应数据这一过程称为一个响应过程，响应过程过程是互斥的。响应模式可以使得多个用户能更充分地利用串口，当多个用户进行相同的数据请求时，可以合成为一个请求，从而较大地提高访问效率。

（2）独占模式，在此模式下，串口为某用户独自占有，一直到其明确地释放了该串口。这一模式的好处是如果需要长时间与串口进行通信，可以避免其它用户的干扰。

3.3 接口模块

接口模块是整个系统中最重要的部分，TCP 数据的接收及串口数据的接收都依赖于它的实现，前文提到了 TCP 以及串口采用统一的接口进行抽象，串口服务器的 TCP 连接同时采用了 C/S 模式和 S/C,也即串口服务器同时以服务器的形式供上位机连接，还能以客户端的方式主动地去连接上位机，这在服务器地 IP 地址为通过 DHCP 自动获取时非常有用，因为此时上位机无从知晓串口服务器的 IP 地址而无法连接。

由于要同时服务于多个用户，同时又要随时地接收来自串口的数据，也就是说串口服务器应该具有良好的并发性，以不至于在多个用户同时访问时产生瓶颈。采用 Select 模型来实现 I/O 的操作[4]，该模型首先使用 FD_SET 将需要侦听的客户 fd，TCP 侦听 fd 以及串口 fd 加入到侦听集当中，然后使用函数 select()侦听，当任何一个 fd 上有数据到来时马上返回，将数据接收完成后放入接收数据队列后返回继续进行侦听，流程如图 3 所示。

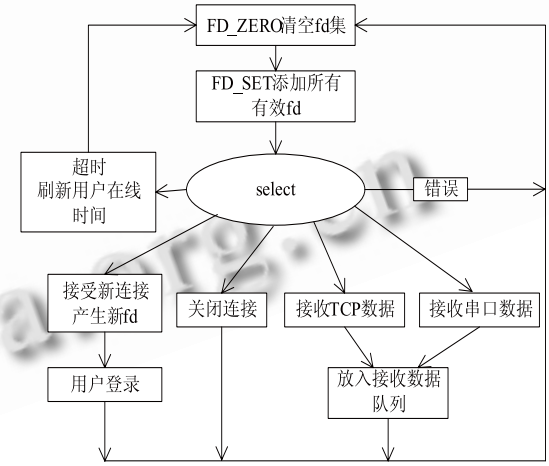

图 3 select 模型流程

3.4 SSL 的扩展

针对 ARM9 的运算能力，只要求上位机对串口服务器的数字证书进行验证，服务器证书名为 server.pem，由 CA 进行签证。CA 可以是正规证书签发机构，也可以是自己建立的非正式签发机构。根据非对称密码学，CA 使用其私钥签名的数据证书只有其公钥才能解密，将其公钥及相关信息做为根证书，只要上位机保证从安全的途径得到根证书，因为只需要一次获取，从而不会出现中间人攻击的情况。

因为 SSL 通信是建立在 TCP 协议基础之上，可以在原 socket 程序代码之上实现 SSL 程序，openSSL 很好地对 SSL 进行了封装[5]，且在 1.0 之后的版本中提供了对 ARM 处理器的支持，使用 openSSL 对原有 TCP 通信代码进行修改即可实现 SSL 安全通信。修改分别针对于 struct Interface 中的 accept、write、read。对于 SSL，它们分别指向 SSLaccept、SSLdisconn、SSLwrite、SSLread，在 SSLaccept 函数中，在原有 TCP 连接建立之后再进行 SSL 协议的握手过程，在此过程中进行服务器数字证书的验证并交换临时密钥，代码如下：

```
clientFd=accept(listenerFd,(struct
   sockaddr *)(&tmpAddr),(socklen_t *)&len);
bio=BIO_new_socket(clientFd,BIO_NOCLOSE);
ssl=SSL_new(ctx);
SSL_set_bio(cli->ssl,cli->sbio,cli->sbio);
//SSL 握手
if(( ret=SSL_accept(cli->ssl)) <= 0 ) {
      sslErr(ssl);
}
```

在 SSLwrite 及 SSLread 函数中可以简单地将原 socket 中的 read 及 write 函数替换为 openSSL 库中的 SSL_write 和 SSL_read 函数。不过对于 SSLread，由于使用了 select 并发模型，只要 fd 上有数据出现 select 函数即马上返回，而此时可能 openSSL 正在进行数据解密操作，SSL_write 函数则返回错误的信息。对此必须对返回的错误进行处理，一直等到解密完成才能读取返回：

```
do {
      ret=SSL_read(ssl,buf, size);
      switch(SSL_get_error(ssl,ret)){
      case SSL_ERROR_NONE:
            goto read_done; //真正读完
      case SSL_ERROR_ZERO_RETURN:
            ret=SSL_shutdown(ssl);
            goto end;
      case SSL_ERROR_WANT_READ:
            read_blocked=1;
            break;
      default:
            DBG("SSL read problem\n");
      }
} while (SSL_pending(ssl)&& !read_blocked);
```

3.5 SSL 的扩展

在 ARM 这种相对于 PC 处理器运算能力弱很多的处理器上，必须选择安全性高且速度有快的加密算法，RC4 是最理想的选择，下表为采用 SSL_RC4_MD5 加密方式的传输与普通 TCP 传输的一组速率对比：

表 1　TCP 和 SSL 传输速率测试(处理器:190MHz, 以太网:100M)

	TCP(MB/S)	SSL（MB/S）
128k	4.7677	2.1864
256k	4.9393	2.2550
512k	5.0352	2.2884
1M	5.1095	2.3051
2M	5.1359	2.3205
4M	5.1980	2.3428
8M	5.2202	2.3463
16M	5.2170	2.3451

从表中可以看到随着传输数据量的增大，传输率逐步趋于稳定，可以推测出为最大的数据传输率，TCP 约为 5.2MB/S，SSL 约为 2.34MB/S。在 SSL 下，若以半双工的方式传输，而根据 串口固定的波特率，可以以“波特率*N<最大数据传输率”的公式求得在此波特率下可支持的串口数，在测试中，以四个串口为例，当四个串口波特率都设为 460800bps 时，仍能正常通信。

4　总结

通过对设计目标的需求分析，结合 ARM9 及 Linux 软硬件平台丰富资源的优势整合，笔者设计的串口服务器实现了多用户多点访问以及安全通信。为串口设备接入互联网提供了更好的选择。

参考文献

1 刘强,崔莉,陈海明.物联网关键技术及应用.计算机科学,2010,37(6):1–4.

2 Atmel Corporation.AT91 ARM Thumb-based Microco ntro llers AT91SAM9261 Preliminary. 2008. http://www. atmel. com/dyn/resources/prod_documents/6062s.pdf

3 梁志刚.基于 AT91SAM9261 的嵌入式 Linux 的移植及其应用研究[硕士博士学位论文].杭州:浙江工业大学,2009.

4 Stevens WR, Fenner B, Rudoff AM. Unix Network Programming, Volume 1: The Sockets Networking API. Addison-Wesley, 2003.

5 Rescorla E. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.