Analysis and transplant of Openwrt system

Gui-Bing[1,a], Zeng-Bi[2,b]

[1] Guangdong University of Technology, Guangzhou 510006, China

[2] Guangdong University of Technology, Guangzhou 510006, China

[a]basb_2008@163.com, [b]272070973@qq.com

Keywords: Openwrt; tiny6410; 360wifi; transplant;

Abstract. This paper studies the organization structure, software architecture and the software compiler configuration process of Openwrt, and the transplant process of which will also be introduced detailedly on tiny6410 development board, and the 360wifi module is added to it; successfully created a WiFi hotspot, and the tiny6410 development board is built to a simple wireless router.

Introduction

Openwrt with a strong network components and scalability is often used for industrial control equipment, telephone, mini robot, routers and VOIP equipment. It is originated in the software system developed by Linksys Company for the home wireless router product of WRT-54 G and is a highly modular and scalable Linux release version. Currently, Openwrt supports a variety of hardware platforms and can not only be widely used in the embedded devices but can also run on the PC platform based on X86. This paper analyzes the system structure of Openwrt which will be transplanted into the tiny6410 development board.

Openwrt system analysis

The analysis of the openwrt source code structure

Openwrt is a depth of customization buildroot system which is a series of collections of Makefile and patches. It can construct and use the corresponding cross compiler tool chain to construct a complete Openwrt embedded system based on Linux automatically. Now the paper makes the following analysis of the important directory and file of Openwrt:

1) Directory structure

gb@gb-virtual-machine:~/openwrt$ tree -L 1 –d

|-- bin *// Directory generated after make, used to store the object files, such as zImage, rootfs, ipk software package*

|-- build_dir *// Directory generated after make, store source files used by compile, build_dir/host directory is temporary folder generated by constructing the target platform tools, build_dir/toolchain-<arch>* is used to compile a specific tool storage architecture, build_dir/ linux-<arch>_<targetboard> is used to store Linux kernel file and to compile the software packages in the kernel.*

|-- dl *// Directory generated after make which is used to store the download source file*

|-- feeds *// Execute the directory generated after ./script/feeds update -a, used to store the software packages downloaded from the server specified by feed.conf file.*

|-- include *// Store make compiler rules and configuration file*

|-- package *// Store with software package Openwrt owns*

|-- scripts *// Store script file used by compile system*

|-- staging_dir *// Directory generated after make, used to store the compiled object code, staging_dir/toolchai-<arch>* directory to store the compiled cross compiler tool chain.*

|-- target *// Mainly related to the specific embedded platform code, target/linux/ kept the Linux core and the embedded platform configuration file, target/imagebuilder holds the tools for firmware package.*

|-- toolchain *// Mainly used to generate the firmware, compile the target platform of cross compiler and C language library*

2) The structure of software package

In the following, the software package structure of dropbearssh server will be used to illustrate how package is organized:

In the following, the software package structure of dropbearssh server will be used to illustrate how package is organized:

// Define the name of software package definition and URL link

```
PKG_NAME:=dropbear
PKG_VERSION:=0.53.1
PKG_RELEASE:=5
PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.gz
PKG_SOURCE_URL:= \
    http://matt.ucc.asn.au/dropbear/releases/ \
    http://www.mirrors.wiretapped.net/security/cryptography/apps/ssh/dropbear/
PKG_MD5SUM:=6b8d901859d9b8a18e2f6bfe0a892a03
define Build/Compile       //Define the compiling rules
    ….
endef
define Package/dropbear/install   //Define the installation rules
    ….
endef
```

The files in package contain the default configuration file and the initialization file, patches is the software patch.

The analysis of Openwrt software architecture

Openwrt is a Linux release version used in embedded system, like other embedded systems, Openwrt also need to have four parts: the boot load programs (bootloader), Linux kernel, file system and application. Bootloader is the first piece of code executed after system startup or reset, it is mainly used to initialize the processor, memory (RAM) and peripherals, and then load the Linux kernel to run memory (RAM) from the non-volatile memory, and give CPU control to the Linux kernel. The Linux kernel needs to mount a file system after the completion of the Linux operating system as the initialization of the root file system (Root Filesystem). The root file system is the core part of Linux system, it can be used as storage for files and data in Linux system, and it usually includes system configuration files and the library needed by software applications. The software architecture of Openwrt can be described in Fig.1.

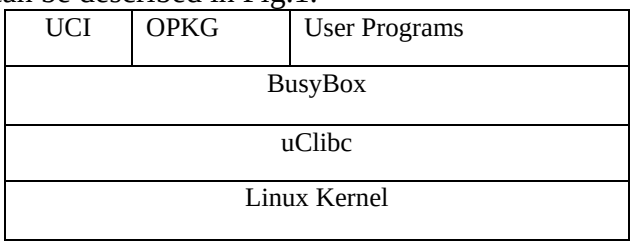

Fig.1: Openwrt architecture

1.UCI (Unified Configuration Interface) is a interface provided by Openwrt for configuration of user space and management system, it is a C language library which can be easily integrated into the user space application and it is to save the system configuration information to the user through the

change of configuration file in the etc/config/ directory. Through the command line or as Luci web interface to modify the UCI configuration file.

2.OPKG is a lightweight tool to download and install the OpenWrt package from a local repository or Internet software warehouse, and it has become the fact standard for embedded open source, similar to apt-get tools in Ubuntu, the OPKG tried to resolve dependencies in the package in the warehouse. If fails, it will report an error and exit stop to install the software package. Input opkg can see the whole help information of the software.

3.User Programs usually exists in the form of .ipk software, which is generated from compile of packages in package directory in Openwrt, through opkginstall xx.ipk, you can install the software.

4.BusyBox is the root file system in Openwrt and the file carrier for Openwrt system to run, BusyBox usually kept the system command, the system configuration file, the library used by application software run and some user's applications.

5.uClibc is a software library Openwrt operation.

6.Linux Kernel is a Linux operating system, mainly responsible for the process management, memory management, file system and device management in system to provide the basic operation environment for the upper application.

Construction of Openwrt system

Openwrt is a Linux release version used for embedded device particular router, the transplant of Openwrt mainly include bootloader transplant, Linux kernel transplant and root file system transplant. Usually, the mirror of embedded Linux system written flash in nonvolatile memory is shown in Fig.2.

Fig.2: The mirror on Openwrt flash

Burning in front of the flash memory is Bootloader; in general, power system Bootloader code in the first run, after the completion of system initialization tasks, from the non-volatile memory (usually Flash or DOC) to copy the Linux kernel to RAM, then jumps to the first instruction of kernel continue to execute, and start the Linux kernel, after the initialization of kernel will start init process, the init kernel process will mount the root file system, and execute the init program in root file system, and then the init program reads the /etc/inittab file, to initialize the system environment according to the set of data in /etc/inittab.

1. Bootloader transplant

The common embedded Linux Bootloader: Blob, Redboot, U-Boot; we select U-Boot which is the most commonly used guiding device in the embedded system, for the compiler environment we use tiny6410 development version of the cross compiler tool chain. As for the transplant of Openwrt, tiny6410 development board and the Samsung SMDK6410 development version only differ from network port and DDR RAM, so basically only modify the network port, the configuration of RAM sequence and the size distribution, the transplant process as follows:

Establish the tiny6410 directory under the board directory, copy the files in SMDK6410 to tiny6410 directory, and change smdk6410.c to tiny6410.c.

1) Modify the network port driver

Define DM9000 network initialization function and register configuration parameter in the tiny6410.c.

```
#define DM9000_Tacs     (0x0)   // 0clk         address set-up
#define DM9000_Tcos     (0x4)   // 4clk         chip selection set-up
#define DM9000_Tacc     (0xE)   // 14clk        access cycle
#define DM9000_Tcoh     (0x1)   // 1clk         chip selection hold
#define DM9000_Tah      (0x4)   // 4clk         address holding time
#define DM9000_Tacp     (0x6)   // 6clk         page mode access cycle
#define DM9000_PMC      (0x0)   // normal(1data)page mode configuration
```

```
// Define DM9000 initialization function
static void dm9000_pre_init(void)
{
    SROM_BW_REG &= ~(0xf << 4);
    SROM_BW_REG |= (1<<7) | (1<<6) | (1<<4);
    SROM_BC1_REG =
((DM9000_Tacs<<28)+(DM9000_Tcos<<24)+(DM9000_Tacc<<16)+(DM9000_Tcoh<<12)+(
DM9000_Tah<<8)+(DM9000_Tacp<<4)+(DM9000_PMC));
}
```

Add DM9000configuration in include/configs/tiny6410.h

```
#define CONFIG_DRIVER_DM9000   1
#define CONFIG_DRIVER_DM9000_NO_EEPROM 1
#define CONFIG_DM9000_USE_16BIT 1
#define CONFIG_DM9000_BASE    0x18000300
#define DM9000_IO CONFIG_DM9000_BASE
#define DM9000_DATA (CONFIG_DM9000_BASE+4)
```

2) Modify the DDR RAMconfiguration parameters

Using 128MB K4X1G163PE-FGC6 DDR memory, make the following settings of MEMORY CONFIGURATION REGISTER according to the tiny6410.

```
#define DMC1_MEM_CFG            ((1<<30) | (2<<15) | (2<<3)| (2<<0))
#define DMC1_MEM_CFG2           0xB41
#define PHYS_SDRAM_1_SIZE       0x8000000 /* 128 MB */
```

3) Modify the machine code

The type of machine with the Linux kernel and u-boot machine types must be consistent, otherwise unable to start. Modify the definition of MACH_TYPE in uboot source code include\configs\Tiny6410.h file: #define MACH_TYPE2521, and modify the definition of MACH_TYPE in linux source code arch\arm\toolsmach-types file:

```
tiny6410MACH_TINY6410TINY64102521
```

4) Set up the linux kernel boot parameters

Tiny6410 DDR of physical memory starting at address 0x50000000, since the start of the MMU virtual memory mapping, 0x50000000 for 0xc0000000, define the boot commands:

```
#define CONFIG_BOOTCOMMAND  "nandread.i c0008000 80000 500000;bootm c0008000"
```

Because the root file system in Linux kernel is stored in the nandflash in the third district, the console using the first serial port, and set the baud rate 115200bps, the bootarg is set to:

```
#define CONFIG_BOOTARGS    "root=/dev/mtdblock2 console=ttySAC0,115200"
```

2. The Openwrt compiler configuration

1) Build compiler environment

// install and compile openwrt basis software installation

gb@gb-virtual-machine:~$ sudo apt-get install libtool autoconf automake bison gcc g++ binutils patch bzip2 flex make gettext unzip libc6 subversion git-core git build-essential libncurses5-dev gawk quilt libz-dev

2) Compile openwrt

//Get the source code from SVN server

gb@gb-virtual-machine:~$ svn co svn://svn.openwrt.org.cn/dreambox/branches/ dreambox-EOL openwrt

//update and install package

```
gb@gb-virtual-machine:~/openwrt$ ./scripts/feeds update –a
gb@gb-virtual-machine:~/openwrt$ ./scripts/feeds install -a
```

//generate a default configuration file

```
gb@gb-virtual-machine:~/openwrt$ makedefconfig
```

//start the Openwrt configuration interface

gb@gb-virtual-machine:~/openwrt$ makemenuconfig

According to the target board, we choose Target System as Samsung S3C64xx, the type of board Target Profile is mini6410 Development board, tiny6410 and mini6410 type is similar, so you can use the mini6410 model.

//compile openwrt

gb@gb-virtual-machine:~/openwrt$ make V=99

3) Transplant Openwrt

a) Add USB driver

In order to add the mini6410 board to support USB driver, we change the target/linux/s3c64xx/ mini6410/profiles/100-min6410-minimal.mk of the mini6410 configuration file, add kmod-usb-core, kmod-usb2and kmod-nls-base.

b) Modify uhttpd server configuration file

Modify the /etc/config/uhttpd configuration files and change the port of listen_http to 80

listlisten_http 80

//Restart the uhttpd server

root@DreamBox:~# /etc/init.d/uhttpd restart

4) Transplant 360WIFI driver

Download the Ralink7601U driver DPA_MT7601U_LinuxAP_ANDROID_20121211.tar.bz2, modify the top layer after decompression.

a) Modify Makefile

Comment PC,open SMDK.

#PLATFORM = PC

PLATFORM = SMDK

Modify the SMDK Linux source code directory and cross compiler directory:

LINUX_SRC=/home/gb/openwrt/build_dir/linux-s3c64xx_mini6410/linux-2.6.36.4

CROSS_COMPILE=/opt/FriendlyARM/toolschain/4.5.1/bin/arm-linux-

b) Add the chip type

As the 360 2generation of WiFi use Ralink7601 chip, add 360 hardware ID in the common /rtusb_dev_id.c in the #ifdef MT7601U.

{USB_DEVICE(0x148f,0x6370)}, /* Ralink 6370 */

{USB_DEVICE(0x148f,0x7601)}, /* MT 6370 */

{USB_DEVICE(0x148f,0x760b)}, /* 360 Wifi */ add this line

5) Modify the network configuration file

By the eth0 network, distribution of this experiment is access directly to the DHCP server assigned IP address, LAN is loaded by the wireless network interface of Ra0 bearing, Ra0 is a Ralink driver after loading the wireless interface name.

So far, you can use terminal to search the WIFI, RT2860AP Wifi is the WiFi hot of this system.

Conclusion

This paper analyzes the Openwrt directory structure, software structure, construction of system, and successfully transplant it intoTiny6410 development board, then compose to a simple router, which lay a foundation for the further study of Openwrt.

References

[1] WIKIPEDIA. Openwrt2015.4 [CP/OL]http://wiki.openwrt.org/doc/start

[2] FriendlyArm2015.4 [CP/OL]http://www.arm9.net/tiny6410.asp

[3] KORSGAARDN P. 2015.4[CP/OL]http://www.buildroot.org/

[4] Edgewall Software. 2015.4,[CP/OL].https://dev.openwrt.org.cn/wiki/DevelopmentIndex

[5] CSDN. 2015.4 [R/OL]http://blog.csdn.net/ofaith12345/article/details/24963457
[6] WIKIPEDIA. Openwrt2015.4 [CP/OL]http://wiki.openwrt.org/doc/recipes/routedap