Advances in Intelligent Systems Research, volume 148
International Conference on Information Technology and Management Engineering (ICITME 2018)

Development and Research of Root File System Based on Cortex-A8 Microprocessor

Li Zhou[1] and Maoqun Yao[2,*]
[1]Institute of Service Engineering, Hangzhou Normal University, Hangzhou Zhejiang 311121, China
[2]Institute of Service Engineering, Hangzhou Normal University, Hangzhou Zhejiang 311121, China
*Corresponding author

***Abstract*—The root file system is an integral part of the embedded Linux system. Busybox is used in the embedded environment to build a root file system, the paper mainly studies how to complete the root file system based on Busybox. This article by root in the ubuntu configuration, compile Busybox and build the root file system necessary files to make the root file system. Finally, successfully mount the experiment by nfs remote mount the root file system in the host ubuntu. The method described in the article can be successfully run in the ARM development board, to provide the necessary environment for the development of embedded systems.**

Keywords—Linux system; root file system; Busybox; Ubuntu; nfs

I. INTRODUCTION

In the Linux system, the file system is a set of software whose function is to manage the sectors of the storage device. When the user accesses a file according to the file name at the application layer, the file system converts the file name to access to the sector number, thereby realizing that all devices in the Linux system are files. The root file system is a file system first. It not only has the function of storing data files, but also the first file system mounted at kernel startup. The kernel code image file is stored in the root file system, and the system boots the startup program. After loading the root file system, some initialization scripts and services are loaded into memory to run. The root file system provides the most basic link libraries, scripts, and special files for Linux system operation. In a complete Linux system, only the kernel itself can not work, like common Linux systems such as ubuntu, redhat, centos, etc. must be in the root file system etc directory under the configuration file / bin, / sbin under the shell directory The commands, as well as the library files in the /lib directory, work together. In other words, the root file system provides the Linux root directory. The common ls, cd, and pwd commands are implemented on the root file system. The root file system also includes the init process, which implements a leap from kernel mode to user mode**[1]**. This article will make a good root file system in ubuntu through NFS remote mount to the s5pv210-based development board, and gives the test results.

II. ROOT FILE SYSTEM

A. Root File System Introduction

The embedded Linux system can be summarized as an application layer and a kernel layer. The application layer is mainly user software. The kernel layer includes a root file system, an operating system, and the underlying hardware.**[2]** The architecture of its components is shown in Figure 1. The specific directories and functions of the Linux root file system are shown in Table 1.

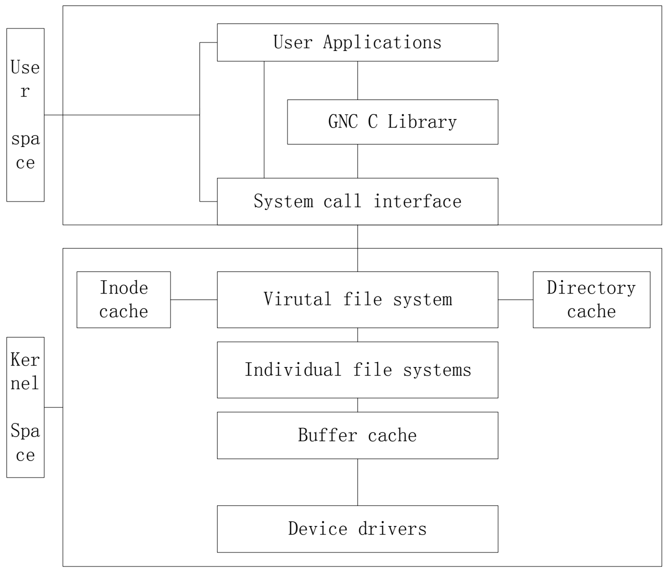


FIGURE I. LINUX SYSTEM ARCHITECTURE

TABLEI. LINUX ROOT FILE SYSTEM DIRECTORY TABLE

/(Root directory)	/bin	Store linux general user command directory
	/sbin	Storage administrator system commands
	/dev	Storage device file
	/etc	Store various configuration files
	/lib	Store shared libraries and loadable drivers
	/root	Root user
	/usr	Store shared, read-only programs and data
	/mnt	Provide users with temporary mounting of other file systems
	/opt	Additional installation software location
	/tmp	The directory where temporary files are stored
	/var	Store variable data, log files
	/media	Storage and mounting equipment
	/proc	The mount point of the file system

B. Build a Root File System

Create an empty root file system file in the host ubuntu and name it rootfs. Then create necessary subdirectories such as bin, sbin, dev, etc, lib, var, etc. in the rootfs directory. The specific operations are as follows[3]:

```
mkdir rootfs
cd rootfs
mkdir bin dev etc proc root sbin sys tmp usr var
```

III. THE TRANSPLANT OF BUSYBOX

Busybox is a big toolbox that integrates many basic commands for Linux systems, such as ls, cd, echo, and so on. The Busybox configuration can be done in menuconfig just like the kernel configuration. Busybox is also an open source project. All source code can be downloaded from the Internet. In this research, Busybox makes its own root file system.

A. Modify Makefile and CROSS_COMPILE

Graphics may be full colour but make sure that they are In the official website to download Busybox1.24.1 to the host ubuntu system, decompression is completed, the first step to modify the Makefile, because the development board uses the cpu Samsung s5pv210 belongs to the Cortex-A8 series architecture, so its ARCH belongs to the arm series, In the ubuntu system to find the cross compiler tool chain installation, copy the relevant path to the Makefile CROSS_COMPILE, the specific code is modified as follows[4,5]:

```
ARCH = arm
CROSS_COMPILE=/usr/local/arm/arm-2009q3/bin//arm-
none-linux-gnueabi-
```

B. Configure Busybox

Bussybox in the menuconfig related configuration, and then compiled, installed into the rootfs directory.

The Busybox runtime needs to be independent of other libraries. When Busybox is configured, Busybox is compiled into a statically linked executable file. Specific steps are as follows[6]:

```
    Busybox Settings--->
        Build Options--->
               [*]Build BusyBox as a static binary(no
shared libs)
# Vi style line editing commands
    Busybox Library Tuning--->
        [*]vi-style line editing commands
        [*]Fancy shell prompts
# Uncheck Simplified modutils, use full tool command
Linux Module Utilities--->
        [ ]Simplified modutils
        [*]insmod
        [*]rmmod
        [*]lsmod
        [*]modprobe
        [*]depmod
# Make sure mdev is supported
    Linux System Utilities--->[*]mdev
        [*]Support /etc/mdev.conf
        [*]Support subdirs/symlinks
        [*]Support regular expressions substitutions when
renaming dev
        [*]Support command execution at device
addition/removal
        [*]Support loading of firmwares
```

IV. BUILD A ROOT FILE SYSTEM

After Busybox is compiled and installed into the specified rootfs directory, the corresponding tool files are generated in the /bin, /sbin, and /usr directories. The necessary configuration files such as inittab, rcS, etc. must also be added to enable the root file to be created. The system works[7].

A. Build Inittab File

Inittab is a runtime configuration file in the /etc directory. When the Linux system is running, it parses the text file and decides how to work according to the contents of the resolution, including the user's run level settings, login system settings, restart settings, etc. Its format is defined in Busybox, and it is parsed in line units. The configuration items for each line of the inittab text file are determined by the four configuration values id: runlevels: action: process. The two configuration values, id and runlevels, can be left blank. Action is a conditional state, and process is the path name of a program that can be executed[8]. That is, the process will be executed when the conditions of the action are satisfied. The build code looks like this:

```
#Run the system script file
::sysinit:/etc/init.d/rcS
s3c2410_serial2::sysinit:/bin/login
::sysinit:/bin/login
::ctrlaltdel:-/sbin/reboot
# Remove all file systems
::shutdown:/bin/umount -a -r
# Restart the init process
::restart:/sbin/init
```

B. Build an rcS File

The rcS file is the most important file in the Linux runtime configuration file. Other configurations are derived from the rcS file, which is located in the /etc/init.d directory. In this file, the PATH environment variable is an environment variable defined inside the Linux system. This ensures that the user can directly use Linux system commands such as ls, cd, and pwd when entering the command line. Set the system run level in the file to single-user mode, ie runlevel=S. Set the file to user-readable and writable mode, umask=022.Use mount-a to mount all filesystems. Of course, all mount points /proc, /sys, /var, /tmp, and /dev must be created in the root directory. The file can also be used to set the host name and host network card ip. The specific operation is shown in the following code[9]:

```
# Set the environment variable
PATH=/sbin:/bin:/usr/sbin:/usr/bin
# Set user mode to single user mode
Runlevel=S
Prevlevel=N
# Set the file permissions to 022 and user default permissions
to 644
Umask 022
# Export environment variables
```

```
Export PATH runlevel prevlevel
#mount all devices
Mount -a
# Generate dev directory related content
Echo /sbin/mdev > /proc/sys/kernel/hotplug
Mdev -s
# Set the host name, here set to zhouli
/bin/hostname -F /etc/sysconfig/HOSTNAME
# Set the host network card ip
Ifconfig eth0 172.17.64.188
```

C. Build a Device Driver File

1) Build fstab file

After the mount point is created in the rootfs directory, the mount-a command in the Busybox parsing the rcS file will look for the file /etc/fstab. This file will list all the files that should be mounted in a certain format. System, build code is as follows[9]:

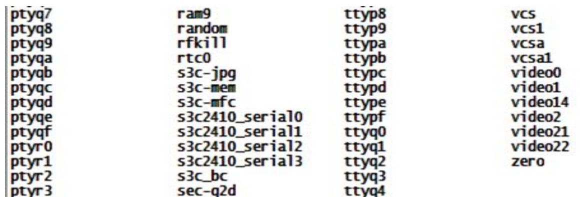
```
# File System Mount Point   Type   Options   Backup Verify Sectors

proc      /proc  proc       defaults  0             0
sysfs     /sys  sysfs       defaults  0             0
tmpfs     /var      tmpfs   defaults   0            0
tmpfs     /tmp  tmpfs  defaults        0            0
tmpfs     /dev  tmpfs       defaults  0             0
```

2) Configure mdev

In Linux system, udev is a device manager. Its main function is to manage device nodes in the dev directory. mdev is a simple udev that comes with Busybox. It is an application layer software, under the /dev directory. Device driver files are generated by medv. When the rcS file does not start mdev, there is no device driver file in the /dev directory. Add related mdev-related configuration in the rcS file, as follows:

```
# Generate dev directory related content
Echo /sbin/mdev > /proc/sys/kernel/hotplug
Mdev –s
```

After restarting the system, use the ls command to view a lot of device driver files generated in the /dev directory, indicating that the device driver file is successfully built, as shown in Figure 2[10]:



```
ptyq7        ram9               ttyp8      vcs
ptyq8        random             ttyp9      vcs1
ptyq9        rfkill             ttypa      vcsa
ptyqa        rtc0               ttypb      vcsa1
ptyqb        s3c-jpg            ttypc      video0
ptyqc        s3c-mem            ttypd      video1
ptyqd        s3c-mfc            ttype      video14
ptyqe        s3c2410_serial0    ttypf      video2
ptyqf        s3c2410_serial1    ttyq0      video21
ptyr0        s3c2410_serial2    ttyq1      video22
ptyr1        s3c2410_serial3    ttyq2      zero
ptyr2        s3c_bc             ttyq3
ptyr3        sec-g2d            ttyq4
ptyr4        sequencer          ttyq5
```

FIGURE II. DEVICE DRIVER FILE MAP

D. Build an rcS File

During the Linux system running, the /etc/profile file is automatically invoked by Busybox. The environment variables set in the profile file affect all users in the Linux system[11]. A typical profile is built in the /etc directory. The code looks like this:

```
# Get the currently logged in user name
USER="`id -un`"
# The current user's login name
LOGNAME=$USER
# basic prompt, # for root user, $ for normal user
PS1='[\u@\h \W]\#'
# Determines which directories the shell will find commands or programs
PATH=$PATH
# Current host name
HOSTNAME=`/bin/hostname`
# Export environment variables
Export USER LOGNAME PS1 PATH
```

E. Set up User Login

1) Login interface settings

There is a user login interface on Linux systems. In this experiment, a simple character interface was created to interact with the user[11]. In the /etc/sysconfig/HOSTNAME file can be used to set the name of the prompt in the development board command line, in the HOSTNAME file to set the prompt name to "zhouli", so that the command line prompt will be displayed in front of [@zhouli] #. Since the configuration item s3c2410_serial2::sysinit:/bin/login has been set in the inittab file, the login will not enter the command line and directly enter the login interface. The specific implementation command is as follows:

```
vi /etc/sysconfig/HOSTNAME
zhouli
```

2) Username and password settings

The files used to describe user names and passwords on Linux systems are passwd and shadow files, both of which are in the /etc directory. The passwd file stores the user's login password, and the shadow file stores the encrypted password. Directly copy the /etc/passwd and /etc/shadow files from the Ubuntu system to the currently created rootfs directory[12]. After the corresponding changes are made, the login user name on the development board is root, and the password is the root user of the ubuntu system. password. The specific order is as follows:

```
cp /etc/passwd  /etc/shadow ./rootfs
```

V. NFS MOUNT

NFS is a network communication protocol consisting of a server and a client. The use of nfs in this production is mainly used as a mount of the root file system. In the host ubuntu system set up a configured nfs server, the kernel running in the development board as a client of nfs. In the host ubuntu's nfs server, export the rootfs folder in the form of a folder. In the development board, you can mount the root file system in this folder and start the kernel on the development board. Starting the kernel in this way is equivalent to the remote mounting of the kernel on the development board to the root file system in the host ubuntu. The specific NFS configuration is as follows:

A. Install nfs

After installing nfs in ubuntu with the sudo apt-get install nfs-kernel-server and sudo apt-get install nfs-common commands, open /etc/exports and add /root/rootfs *(rw,sync, No_ root_ squash, no_ subtree_ check), that is, the root file system in /root/rootfs is mounted on the development board. You also need to use chmod 777 -R /root/rootfs to modify the permissions so that the development board can directly execute the ubuntu program. Sudo /etc/init.d/nfs-kernel-server restart Restart the nfs service.

B. Nfs Configuration

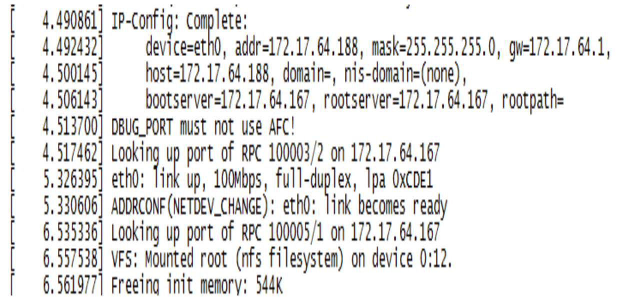
#IP to support kernel configuration
[*] Networking support --->
Networking options --->
[*]IP: kernel level autoconfiguration
# File system supports nfs client and making root file system with nfs
File systems --->
[*] Network File Systems --->
--- Network File Systems
<*>NFS client support
[*]NFS client support for NFS version 3
[*]NFS client support for the NFSv3 ACL protocol extension
[*]NFS client support for NFS version 4 (EXPERIMENTAL)
[*]NFS client support for NFSv4.1 (DEVELOPER ONLY)
[*]Root file system on NFS

C. Set Uboot Bootargs

When uboot starts, it will pass parameters to the kernel via bootargs, set uboot's bootargs such as the path of the root directory, host IP, development board IP, gateway, serial port, baud rate, etc. to support the nfs way to mount the root file in ubuntu. System, the specific settings are as follows:

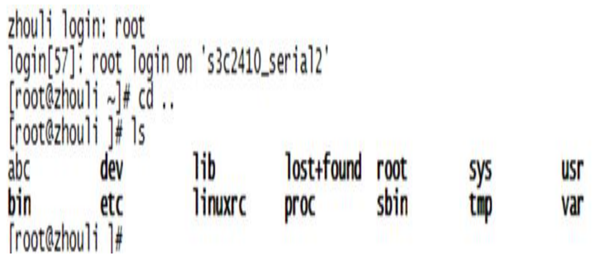
```
setenv bootargs   root=/dev/nfs
nfsroot=172.17.64.141:/root/porting_x210/rootfs
ip=172.17.64.188:172.17.64.167:172.17.64.1:255.255.255.0::
eth0:off init=/linuxrc console=ttySAC2,115200
```

After the setup is complete, start the Linux system on the development board, log in as the root user, and successfully enter the shell command line. You can use the shell command to prove that the nfs mounts the root file system remotely. Some of the experimental processes are shown in Figure 3. The result is shown in Figure 4:



```
[    4.490861] IP-Config: Complete:
[    4.492432]      device=eth0, addr=172.17.64.188, mask=255.255.255.0, gw=172.17.64.1,
[    4.500145]      host=172.17.64.188, domain=, nis-domain=(none),
[    4.506143]      bootserver=172.17.64.167, rootserver=172.17.64.167, rootpath=
[    4.513700] DBUG_PORT must not use AFC!
[    4.517462] Looking up port of RPC 100003/2 on 172.17.64.167
[    5.326395] eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1
[    5.330606] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[    6.535336] Looking up port of RPC 100005/1 on 172.17.64.167
[    6.557538] VFS: Mounted root (nfs filesystem) on device 0:12.
[    6.561977] Freeing init memory: 544K
```

FIGURE III. NFS MOUNT STARTUP PROCESS DIAGRAM



```
zhouli login: root
login[57]: root login on 's3c2410_serial2'
[root@zhouli ~]# cd ..
[root@zhouli ]# ls
abc      dev      lib      lost+found root     sys      usr
bin      etc      linuxrc  proc       sbin     tmp      var
[root@zhouli ]#
```

FIGURE IV. SUCCESSFUL MOUNTING PHENOMENON

VI. CONCLUSION

In an embedded Linux system, a good root file system can maximize resource utilization and improve overall system performance. This article begins with an empty folder, describes in detail the composition, transplantation, production, and mounting of the root file system. It is believed that there will be great help for the study of the embedded Linux file system.

This article innovation: the general root file system is built on the original file to cut, in this article, the use of Busybox plus necessary files to create a new root file system, in this process elaborated inittab, rcS, etc. The composition of important files, when testing the experimental results, uses nfs to mount the root file system remotely. Finally, the root file system that has been successfully created is successfully run on the development board.

ACKNOWLEDGEMENTS

Current-mode CMOS digital and integrated circuit theory and technology research (61771179), Research on the design of threshold arithmetic algebraic system and current-mode CMOS circuit (LY15F010011)

REFERENCES

[1] Matt Welsh & Lar Kaufman. "The Definitive Guide to Linux" [M]. Hong Feng translated. 3rd edition. China Electronic Publishing, 2000.

[2] Barr. Linux File System [M]. Tsinghua University Press, 2003.

[3] Zou Sibiao. Embedded Linux Design and Application [M]. Tsinghua University Press, 2002.

[4] Ma Lijie. Embedded Linux Application Programming [M]. Beijing Institute of Technology Press, 2016.

[5] Yan Haitao, Nie Hongyi, Chen Qingwen. The Creation and Porting of Root File System Based on Embedded Linux[J]. Automation Technology and Applications, 2014, 33(12):21-24.

[6] Peng Hao, Gong Jie, Qin Jianmin. Embedded Linux Root File System Based on S3C2440[J]. Electronic Design Engineering, 2010, 18(6):20-22.

[7] Liu Ergang. A Method of Constructing Embedded Linux Root File System [J]. Electronic Design Engineering, 2016, 24(9):160-162.

[8] Xiong Xingxing, He Yueshun. U-boot Analysis and Porting Based on S5PV210[J]. Journal of Computer Systems, 2015, 24(1):199-205.

[9] Du Haixing. Embedded Bootloader Analysis and Porting Based on ARM[J]. Microcomputer Information, 2010, 26(29):58-59+57.

[10] Blum R. Linux Command Line and Shell Scripting Bible[M]. John Wiley & Sons, 2015.

[11] Nam S, Yoon S K, Kim S D. Fast bootstrapping method for the memory-disk integrated memory system[C]// Ieee/acis, International Conference on Computer and Information Science. IEEE, 2015:167-172.

[12] Wang L. Realization of U-Boot booting through NAND Flash[J]. Electronic Design Engineering, 2010.