Operating Systems And Applications For Embedded Systems

Root Filesystem

Plan

Useful System

- init: The program that starts everything off, usually by running a series of scripts.
- shell: Needed to give you a command prompt but, more importantly, to run the shell scripts called by init and other programs.
- daemons: Various server programs, started by init.
- libraries: Usually, the programs mentioned so far are linked with shared libraries which must be present in the root filesystem.
- Configuration files: The configuration for init and other daemons is stored in a series of ASCII text files, usually in the /etc directory.
- Device nodes: The special files that give access to various device drivers.
- /proc and /sys: Two pseudo filesystems that represent kernel data structures as a hierarchy of directories and files. Many programs and library functions read these files.
- kernel modules: If you have configured some parts of your kernel to be modules, they will be here, usually in /lib/modules/[kernel version].

Filesystem Hierarchy Standard (FHS)

- /bin: programs essential for all users
- /dev: device nodes and other special files
- /etc: system configuration
- /lib: essential shared libraries, for example, those that make up the C library
- /proc: the proc filesystem
- /sbin: programs essential to the system administrator
- /sys: the sysfs filesystem
- /tmp: a place to put temporary or volatile files
- /usr: as a minimum, this should contain the directories /usr/bin, /usr/lib and /usr/sbin, which contain additional programs, libraries, and system administrator utilities
- /var: a hierarchy of files and directories that may be modified at runtime, for example, log messages, some of which must be retained after boot

Staging directory

1. mkdir /rootfs
2. cd /rootfs
3. mkdir bin dev etc home lib proc sbin sys tmp usr var
4. mkdir usr/bin usr/lib usr/sbin
5. mkdir var/log

The init program

The init is the frst program to be run and so has PID 1. It runs as the root user and so has maximum access to system resources. Usually, it runs shell scripts which start daemons: a daemon is a program that runs in the background with no connection to a terminal, in other places it would be called a server program.

Shell

- bash: is the big beast that we all know and love from desktop Linux. It is a superset of the Unix Bourne shell, with many extensions or bashisms.
- ash: also based on the Bourne shell, and has a long history with the BSD variants of Unix. Busybox has a version of ash which has been extended to make it more compatible with bash. It is much smaller than bash and hence is a very popular choice for embedded systems.
- hush: is a very small shell that we briefly looked at in the chapter on bootloaders. It is useful on devices with very little memory. There is a version in BusyBox.

Utilities

To make a shell useful, you need the utility programs that the Unix command-line is based on. Even for a basic root flesystem, there are approximately 50 utilities.

BusyBox

1. git clone git://busybox.net/busybox.git
2. cd busybox
3. make distclean
4. make defconfig
5. make install

ToyBox

ToyBox has the same aim as BusyBox, but with more emphasis on complying with standards, especially POSIX-2008 and LSB 4.1, and less on compatibility with GNU extensions to those standards. ToyBox is smaller than BusyBox, partly because it implements fewer applets.

Libraries

```
cd /rootfs
arm-cortex_a8-linux-gnueabihf-readelf -a bin/busybox | grep "program interpreter"
[Requesting program interpreter: /lib/ld-linux-armhf.so.3]
arm-cortex_a8-linux-gnueabihf-readelf -a bin/busybox | grep Śhared library"
0x00000001 (NEEDED) Shared library: [libm.so.6]
0x00000001 (NEEDED) Shared library: [libc.so.6]
export SYSROOT=‘arm-cortex_a8-linux-gnueabihf-gcc -print sysroot‘
cd /rootfs
cp -a $SYSROOT/lib/ld-linux-armhf.so.3 lib
cp -a $SYSROOT/lib/ld-2.19.so lib
cp -a $SYSROOT/lib/libc.so.6 lib
cp -a $SYSROOT/lib/libc-2.19.so lib
cp -a $SYSROOT/lib/libm.so.6 lib
cp -a $SYSROOT/lib/libm-2.19.so lib
```

Reducing size by stripping

file rootfs/lib/libc-2.19.so
rootfs/lib/libc-2.19.so: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 3.15.4, not stripped
ls -og rootfs/lib/libc-2.19.so
-rwxrwxr-x 1 1547371 Feb 5 10:18 rootfs/lib/libc-2.19.so
arm-cortex_a8-linux-gnueabi-strip rootfs/lib/libc-2.19.so
file rootfs/lib/libc-2.19.so
rootfs/lib/libc-2.19.so: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 3.15.4, stripped
ls -l rootfs/lib/libc-2.19.so
-rwxrwxr-x 1 chris chris 1226024 Feb 5 10:19 rootfs/lib/libc-2.19.so
ls -og rootfs/lib/libc-2.19.so
-rwxrwxr-x 1 1226024 Feb 5 10:19 rootfs/lib/libc-2.19.so
strip –strip-unneeded <module name>

Device nodes

Device nodes are created using the program mknod (short for make node):

```
mknod <name> <type> <major> <minor>
cd  /rootfs
sudo mknod -m 666 dev/null c 1 3
sudo mknod -m 600 dev/console c 5 1
ls -l dev
total 0
crw------- 1 root root 5, 1 Oct 28 11:37 console
crw-rw-rw- 1 root root 1, 3 Oct 28 11:37 null
```

The proc and sysfs flesystems

proc and sysfs should be mounted on the directories /proc and /sys:
mount -t proc proc /proc
mount -t sysfs sysfs /sys

Mounting flesystems

mount [-t vfstype] [-o options] device directory
mount -t ext4 /dev/mmcblk0p1 /mnt

Additional reading

1. ramdisk: a filesystem image that is loaded into RAM by the bootloader. Ramdisks are easy to create and have no dependencies on mass storage drivers. They can be used in fall-back maintenance mode when the main root filesystem needs updating. They can even be used as the main root filesystem in small embedded devices and, of course, as the early user space in mainstream Linux distributions. A compressed ramdisk uses the minimum amount of storage but still consumes RAM. The contents are volatile so you need another storage type to store permanent data such as configuration parameters.
2. disk image: a copy of the root filesystem formatted and ready to be loaded onto a mass storage device on the target. For example, it could be an image in ext4 format ready to be copied onto an SD card, or it could be in jffs2 format ready to be loaded into flash memory via the bootloader. Creating a disk image is probably the most common option.
3. network filesystem: the staging directory can be exported to the network via an NFS server and mounted by the target at boot-time. This is often done during the development phase in preference to repeated cycles of creating a disk image and reloading it onto the mass storage device, which is quite a slow process.

Standalone ramdisk

```
cd  /rootfs find . | cpio -H newc -ov –owner root:root > ../initramfs.cpio
cd ..
gzip initramfs.cpio
mkimage -A arm -O linux -T ramdisk -d initramfs.cpio.gz uRamdisk
```

Minimizing size

- Make the kernel smaller by leaving out drivers and functions you don't need
- Make BusyBox smaller by leaving out utilities you don't need
- Use uClibc or musl libc in place of glibc
- Compile BusyBox statically

Booting with QEMU

cd /rootfs
QEMU_AUDIO_DRV=none qemu-system-arm -m 256M -nographic -M vexpress-a9 -kernel zImage -append ćonsole=ttyAMA0 rdinit=/bin/sh-dtb vexpress-v2p-ca9.dtb -initrd initramfs.cpio.gz

Additional reading

- Filesystem Hierarchy Standard, currently at version 3.0 available at http://refspecs.linuxfoundation.org/fhs.shtml.
- ramfs, rootfs and initramfs , Rob Landley, October 17, 2005, which is part of the Linux source code available at Documentation/filesystems/ramfsrootfs-initramfs.txt.

References

C. Simmonds.
Mastering Embedded Linux Programming.
Packt Publishing, 2015.

The End