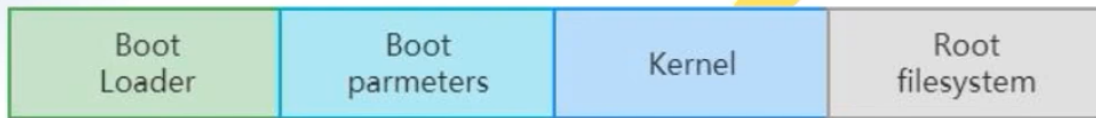


嵌入式Linux系统组成

一个完整的嵌入式Linux系统，包含uboot，Linux内核，根文件系统三个部分。其启动顺序为，在系统刚一上电的时候，先执行uboot，由uboot引导Linux内核，内核启动成功以后挂载根文件系统。从这一期开始，我们来学习根文件系统的相关知识。



什么是文件系统（组织和管理系统文件）

什么是文件系统？

讲什么是文件系统，我们就可以将他分成俩个部分，一部分是文件，一部分是系统。那文件系统是不是可以理解成是用来组织和管理文件的一个系统呢。有了文件系统以后，就可以轻松的操作存储在存储介质的文件。比如删除一个文件，新增一个文件等。

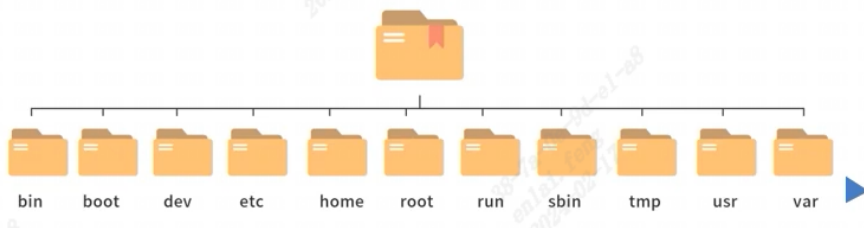
文件系统的格式（类型）有很多种，如fat32，ext2，ext3，ntfs等。

什么是根文件系统

什么是根文件系统呢？

根文件系统也是文件系统。何为根呢？根是不是就是最顶端呀。最顶端称之为根。在Linux上用“/”来表示。然后将一个按照特定目录组成的文件系统挂载到根上。那这个文件系统就叫做根文件系统。

所以根文件系统是一个挂载在Linux系统根目录下的，有特定目录组成的文件系统。



根文件系统目录介绍

根文件系统目录介绍

1 /bin目录

该目录下主要是存放Linux命令，这些可以被root和普通用户所使用，/bin目录下常用的命令有：cat、chgrp、chmod、cp、ls、sh、kill、mount、umount、mkdir等。

2 /sbin目录

该目录下存放的是系统命令，只有系统管理员（俗称最高权限的 root）能够使用的命令，除此之外系统命令还可以存放在/usr/sbin，/usr/local/sbin 目录下。

/sbin 目录下常用的命令有：shutdown、reboot、fdisk、fsck、init等。

3 /dev目录

该目录下存放的是设备接口的文件，设备文件是 Linux 中特有的文件类型，在 Linux 系统下，一些设备文件。所以以文件的方式访问各种设备，即通过读写某个设备文件操作某个具体硬件。比如通过“dev/ttySAC0”文件可以操作串口0等。

4 /etc目录

该目录下存放着系统主要的配置文件。

5 /lib目录

Lib是library的简称，也就是库的意思，因此此目录下存放着Linux所必须的库文件。

6 /home目录

系统默认的用户文件夹，它是可选的，对于每个普通用户，在/home目录下都有一个以用户名命名的子目录，里面存放用户相关的配置文件。

7 /root目录

系统管理员（root）的主文件夹，即是根用户的目录，与此对应，普通用户的目录是/home 下的某个子目录。

8 /usr目录

usr是unix shared resources(共享资源)的缩写，用来存放用户的应用程序和文件。

9 /var目录

var是variable(变量)的缩写，/var目录中存放可变的数据，如日志文件。

10 /proc目录

这是一个空目录，常作为 proc 文件系统的挂接点。

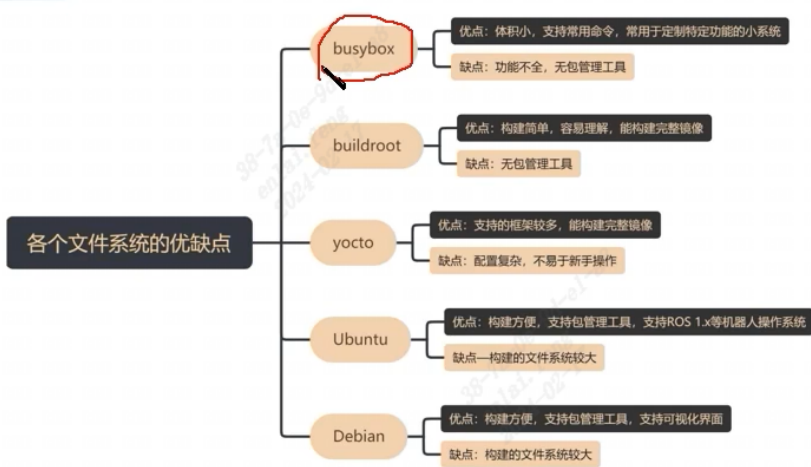
11 /mnt目录

用于临时挂载某个文件系统的挂接点，通常是空目录，也可以在里面创建一引起空的子目录，比如/mnt/cdram /mnt/hda1。用来临时挂载光盘、移动存储设备等。

12 /tmp目录

用于存放临时文件，通常是空目录，一些需要生成临时文件的程序用到的/tmp 目录下。

最简单的根文件系统一般使用 busybox 工具制作。除此之外，还可以使用 buildroot 工具和 yocto 工具构建的文件系统。当然也可以直接使用发行版 Linux 系统，如 Ubuntu 系统，Debian 系统。



busybox

BusyBox 是一个将多个 UNIX 命令和工具集成到一个单一可执行文件中的软件。它被设计用于嵌入式系统，以减少磁盘占用和内存使用。BusyBox 提供了一个精简的环境，包含了一些最常用的命令行工具和工具程序，如 `ls`、`cp`、`mv`、`mount`、`telnet`、`httpd`、`init` 等。

BusyBox 可以配置为包含一组特定的工具，这取决于目标系统的需求和资源限制。这种配置性使得 BusyBox 非常适合用于资源受限的嵌入式系统。

在嵌入式 Linux 系统中，BusyBox 通常用作：

- **命令行接口**：提供标准的 Unix 命令行工具，用于系统管理和调试。
- **系统初始化**：作为 `init` 程序，负责启动系统并运行启动脚本。
- **替代完整的 GNU 工具链**：在资源受限的系统中，BusyBox 提供了一个比完整 GNU 工具链更轻量级的解决方案。

BusyBox 的安装和配置通常涉及以下步骤：

1. **配置**：可以通过图形化配置工具（如 `make menuconfig`）或手动编辑 `.config` 文件来选择要包含在 BusyBox 可执行文件中的命令和功能。
2. **编译**：使用交叉编译工具链编译 BusyBox，生成适用于目标嵌入式系统的二进制文件。
3. **安装**：将编译好的 BusyBox 二进制文件复制到目标系统的文件系统适当位置。
4. **创建符号链接**：为 BusyBox 可执行文件创建一组符号链接，每个链接对应于 BusyBox 支持的命令。这样，当用户尝试执行任何这些命令时，实际上都是执行的 BusyBox 可执行文件。

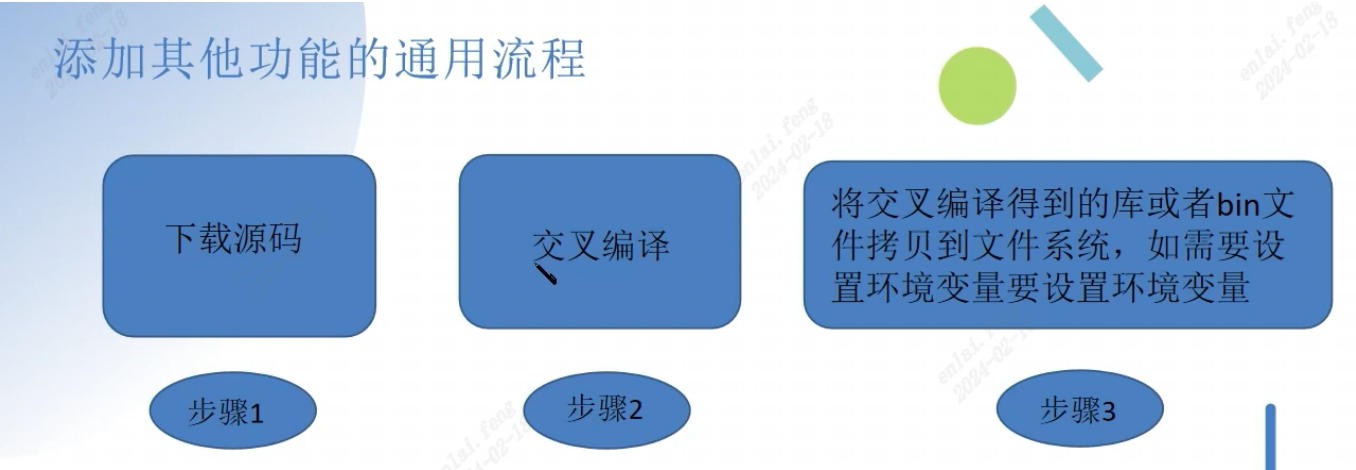
例如，如果你有一个名为 `busybox` 的可执行文件，你可以为 `ls` 命令创建一个符号链接，如下所示：

```
ln -s /bin/busybox /bin/ls
```

当你在命令行中输入 `ls` 时，系统会调用 `/bin/busybox`，并传递 `ls` 作为参数，BusyBox 会识别这个参数并执行相应的 `ls` 命令功能。

busybox打包

```
topeet@ubuntu:~/work/root$ mkdir rootfs
topeet@ubuntu:~/work/root$
topeet@ubuntu:~/work/root$ dd if=/dev/zero of=rootfs.img bs=1M count=300
300+0 records in
300+0 records out
314572800 bytes (315 MB, 300 MiB) copied, 0.669609 s, 470 MB/s
topeet@ubuntu:~/work/root$ ls
busybox-1.33.1 busybox-1.33.1.tar.bz2 rootfs rootfs.img sys sys.tar.gz
topeet@ubuntu:~/work/root$ du sys -h
4.0K sys/var
4.0K sys/dev
12K sys/etc/init.d
12K sys/etc/hotplug
8.0K sys/etc/rc.d
60K sys/etc
4.0K sys/proc
4.0K sys/sys
42M sys/lib/debug
211M sys/lib
1020K sys/bin
4.0K sys/sbin
4.0K sys/mnt
4.0K sys/usr/bin
4.0K sys/usr/sbin
12K sys/usr
4.0K sys/tmp
212M sys
topeet@ubuntu:~/work/root$
```



交叉编译工具

交叉编译工具（Cross-Compiler）是一种编译器，能够为与编译器本身运行的平台（宿主系统）不同的平台（目标系统）生成可执行代码。这是在开发嵌入式系统或操作系统时常用的工具，因为这些系统的硬件平台（目标平台）通常与开发者使用的硬件（宿主平台）不同。

例如，如果你在一台 x86 架构的电脑上（宿主系统）开发一个针对 ARM 架构的嵌入式设备（目标系统）的软件，你会需要一个 x86 到 ARM 的交叉编译器。

交叉编译工具链的组成

一个典型的交叉编译工具链包括：

- **交叉编译器 (Cross-Compiler)**：将源代码编译成目标平台的机器代码。
- **交叉汇编器 (Cross-Assembler)**：将汇编语言转换为目标平台的机器代码。
- **链接器 (Linker)**：将多个对象文件（编译器和汇编器生成的输出）链接成一个单一的可执行文件，解决符号引用和地址分配。
- **库 (Libraries)**：目标平台的标准库和其他库文件，编译器和链接器会使用这些库来编译和链接程序。
- **调试器 (Debugger)**：用于目标平台的调试工具，可以远程调试目标系统上运行的程序。

常见的交叉编译工具链

- **GNU Compiler Collection (GCC)**：提供了广泛支持的交叉编译器，可以编译 C、C++、Java、Fortran、Ada 等语言。
- **Clang/LLVM**：一个模块化和可重用的编译器和工具链技术的集合，支持多种编程语言。
- **CodeSourcery/Mentor Graphics Toolchain**：基于 GCC，但进行了优化和增强，特别是针对嵌入式系统。
- **Yocto Project**：一个协作项目，提供了一个框架来帮助开发者创建定制的 Linux 分发版，适用于嵌入式设备，包括交叉编译工具链。
- **crosstool-NG**：一个交叉编译工具链生成器，可以用来构建交叉编译工具链。

设置交叉编译环境

为了设置交叉编译环境，开发者需要：

1. **选择或构建交叉编译工具链**：可以使用预先构建的工具链，或者使用工具如 crosstool-NG 自己构建。
2. **配置环境变量**：如 `PATH`，以便在命令行中方便地调用交叉编译器和工具。
3. **配置编译系统**：确保构建系统（如 Makefile 或构建脚本）使用正确的编译器和工具链路径。

交叉编译的挑战

交叉编译可能会遇到的挑战包括：

- **库依赖性**：确保目标系统上有正确版本的库。
- **头文件**：需要目标平台的正确头文件来编译程序。
- **二进制兼容性**：编译出的程序必须与目标硬件兼容。
- **调试困难**：可能需要特殊的调试工具或在目标硬件上运行调试器。

交叉编译是嵌入式系统开发的一个重要方面，因为它使得开发者能够在功能强大的宿主机上进行**软件开发和编译**，同时生成适用于资源受限的**目标硬件**的代码。

Buildroot

Buildroot 是一个开源的、简化嵌入式 Linux 系统开发的工具，它使用 makefile 和 kconfig（与 Linux 内核配置系统相同的配置系统）来自动化构建过程。Buildroot 能够生成交叉编译工具链、根文件系统、Linux 内核映像以及引导加载程序，并且可以为不同的目标架构进行配置和编译。

Buildroot 的主要优点是它的简单性和配置的灵活性。使用 Buildroot，开发者可以快速地为嵌入式系统生成轻量级、定制化的 Linux 环境。

Buildroot 的主要特点：

- **简化的配置**：通过图形界面（`make menuconfig`）、命令行界面（`make xconfig` 或 `make nconfig`）或直接编辑配置文件来选择所需的软件包和系统配置。
- **自动生成工具链**：自动下载、配置、编译并安装交叉编译工具链。
- **软件包管理**：包含数千个可用的软件包，可以通过 Buildroot 的配置系统选择并配置这些软件包。
- **文件系统生成**：支持多种文件系统类型，如 ext2/3/4、btrfs、squashfs、jffs2 等，可以根据需要生成镜像文件。
- **引导加载器支持**：支持多种引导加载器，如 U-Boot、GRUB、Barebox 等。
- **定制化**：允许开发者添加定制的软件包和补丁，以及调整内核和系统设置。
- **轻量级**：生成的系统非常适合资源受限的嵌入式设备。

使用 Buildroot 的步骤：

1. **下载 Buildroot**：从官方网站下载最新的 Buildroot 源代码。
2. **配置 Buildroot**：运行 `make menuconfig`，选择目标架构、工具链、内核配置、软件包和文件系统选项等。
3. **构建系统**：运行 `make` 命令开始构建过程。Buildroot 会下载源代码，构建交叉编译工具链，编译内核，构建选定的软件包，并生成根文件系统镜像。
4. **部署和测试**：将构建的系统镜像烧录到目标设备（如 SD 卡、NAND 闪存等），然后在目标硬件上启动并测试。

Buildroot 的挑战：

- **学习曲线**：新用户可能需要一些时间来熟悉 Buildroot 的配置和使用。
- **定制化**：虽然 Buildroot 提供了大量的配置选项，但对于非常特殊的定制需求，用户可能需要编写额外的 makefile 或补丁。
- **软件包更新**：Buildroot 社区提供软件包更新，但用户可能需要手动更新特定软件包或等待 Buildroot 社区更新。

Buildroot 是一个强大的工具，特别适合于那些需要快速、高效地为嵌入式设备生成定制 Linux 系统的场景。

文件系统构建

busybox

优点

- 1.构建的文件系统比较小
- 2.非常灵活
- 3.是一个快捷的构建文件系统的方法
- 4.非常好的学习文件系统构建的工具

缺点

- 1.编译出来的文件系统不完善，需要自行添加。比较麻烦并且对能力要求比较高
- 2.功能单一，不适合项目快速开发

buildroot

优点

- 1.使用起来非常方便，效率高
- 2.功能强大，不光可以构建文件系统，也可以构建内核和uboot

缺点

- 1.对网络有要求，编译过程中需要连接网络，并且有些东西需要科学上网
- 2.隐藏了很多细节，对于想深入了解文件系统构建过程的学习用户不友好

其他文件系统构建

yocto

Linux的发行版

- ubuntu
- debian
- Openwrt