从零开始为 RISC-V 构建一个 Linux 系统

第 3 章 构建 Linux 系统之前的准备工作

汪辰

目录

01 需要制作哪些东西?

02 构建环境准备

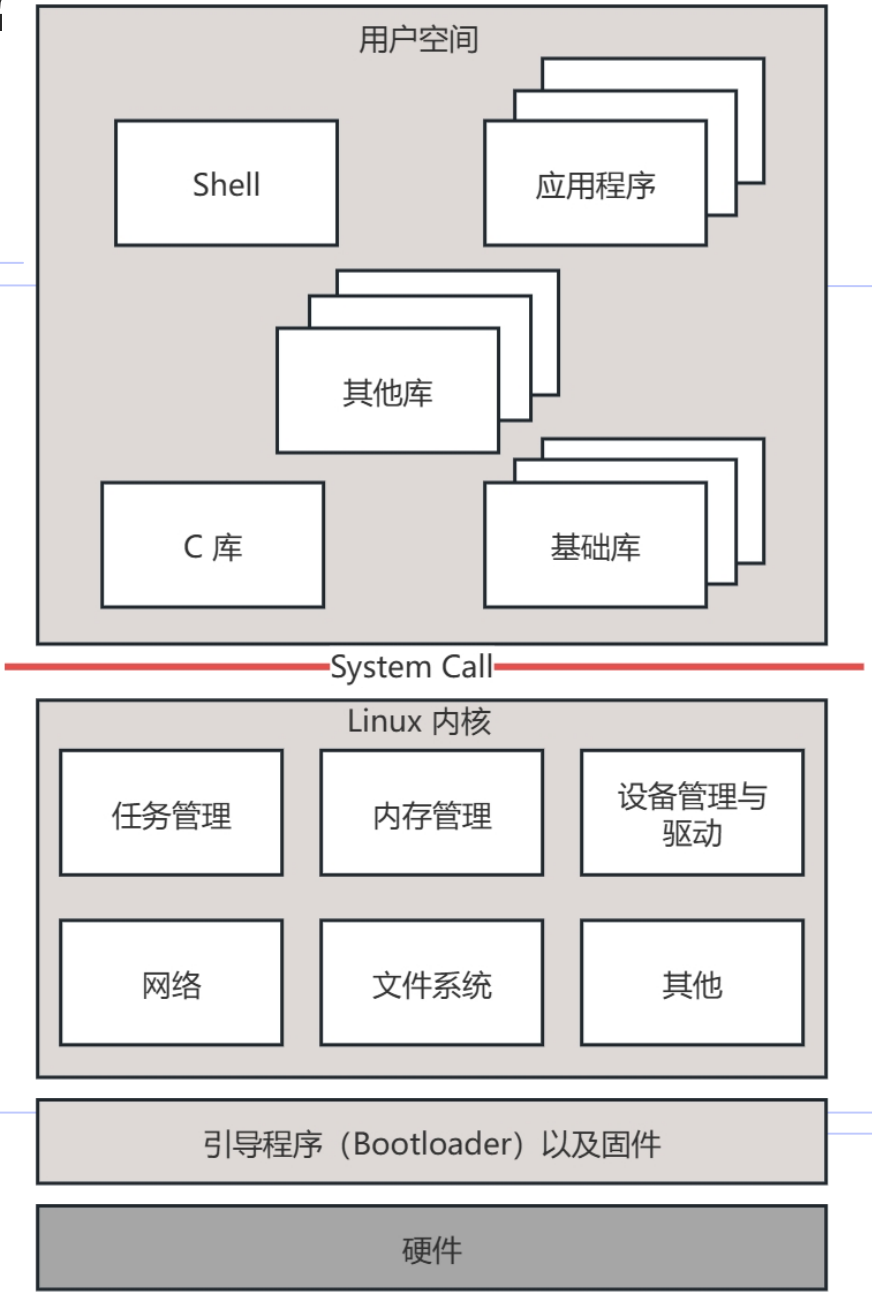
03 构建目录安排

04 构建流程概览

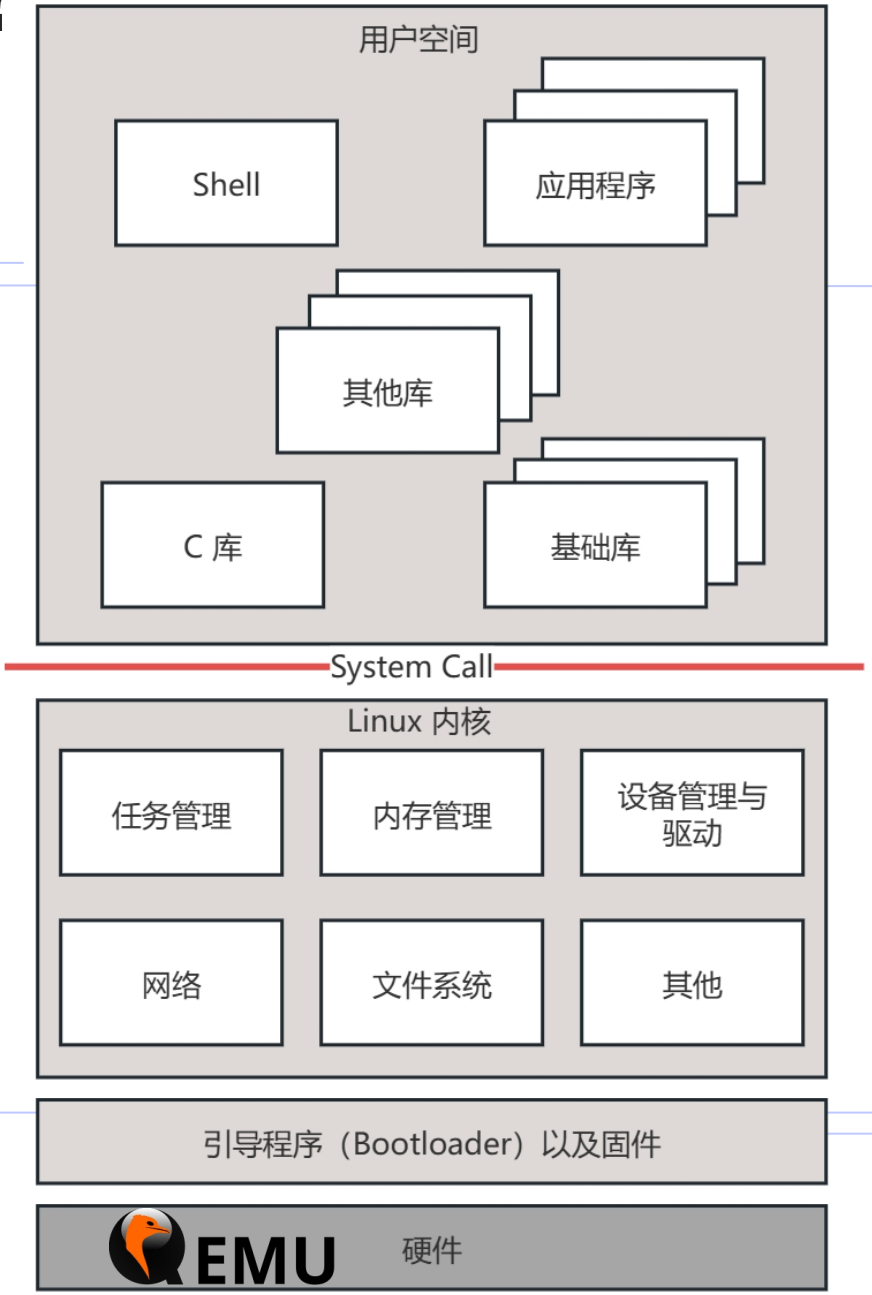
01

需要制作哪些东西?

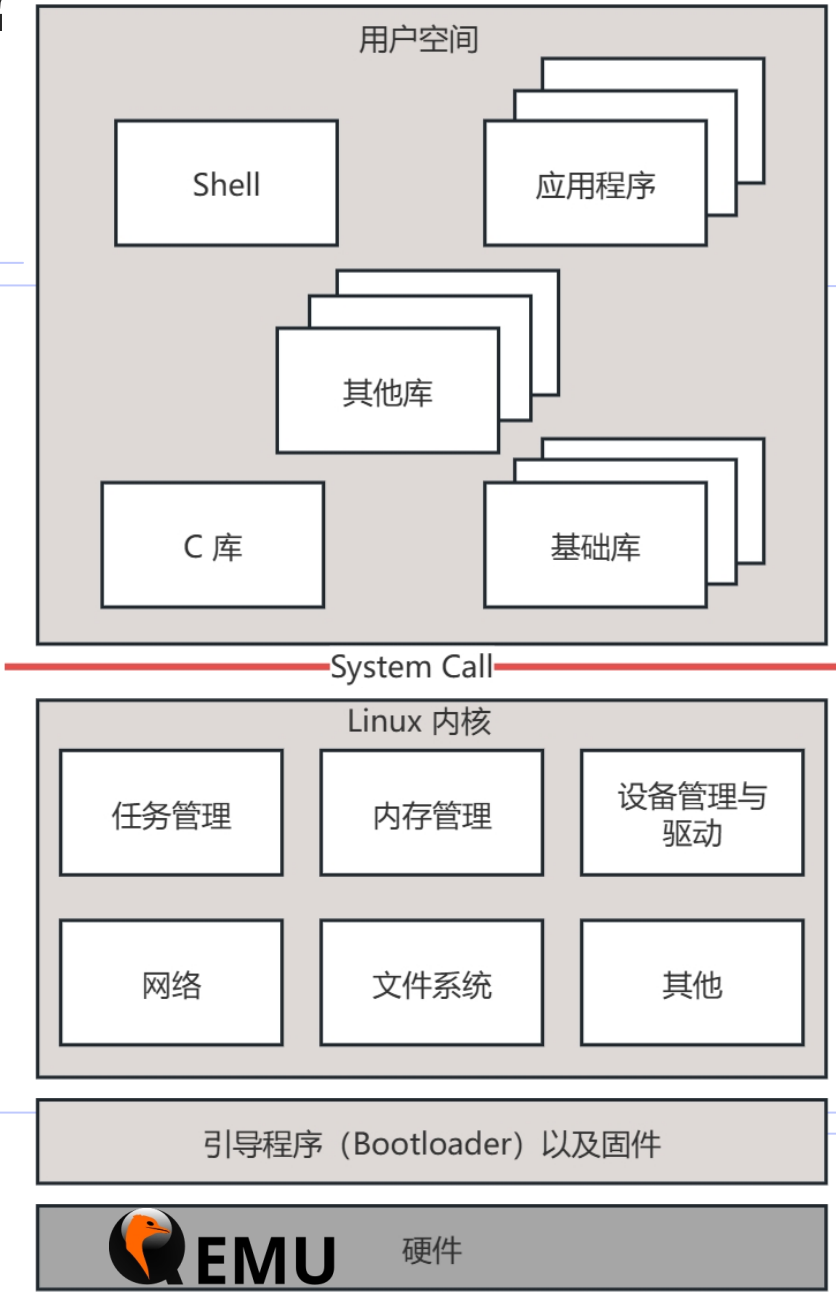
需要制作哪些东西？


用户空间
Shell
应用程序
其他库
C 库
基础库
System Call
Linux 内核
任务管理
内存管理
设备管理与驱动
网络
文件系统
其他
引导程序（Bootloader）以及固件
硬件

需要制作哪些东西？


用户空间
Shell
应用程序
其他库
C 库
基础库
System Call
Linux 内核
任务管理
内存管理
设备管理与驱动
网络
文件系统
其他
引导程序（Bootloader）以及固件
QEMU
硬件

需要制作哪些东西？

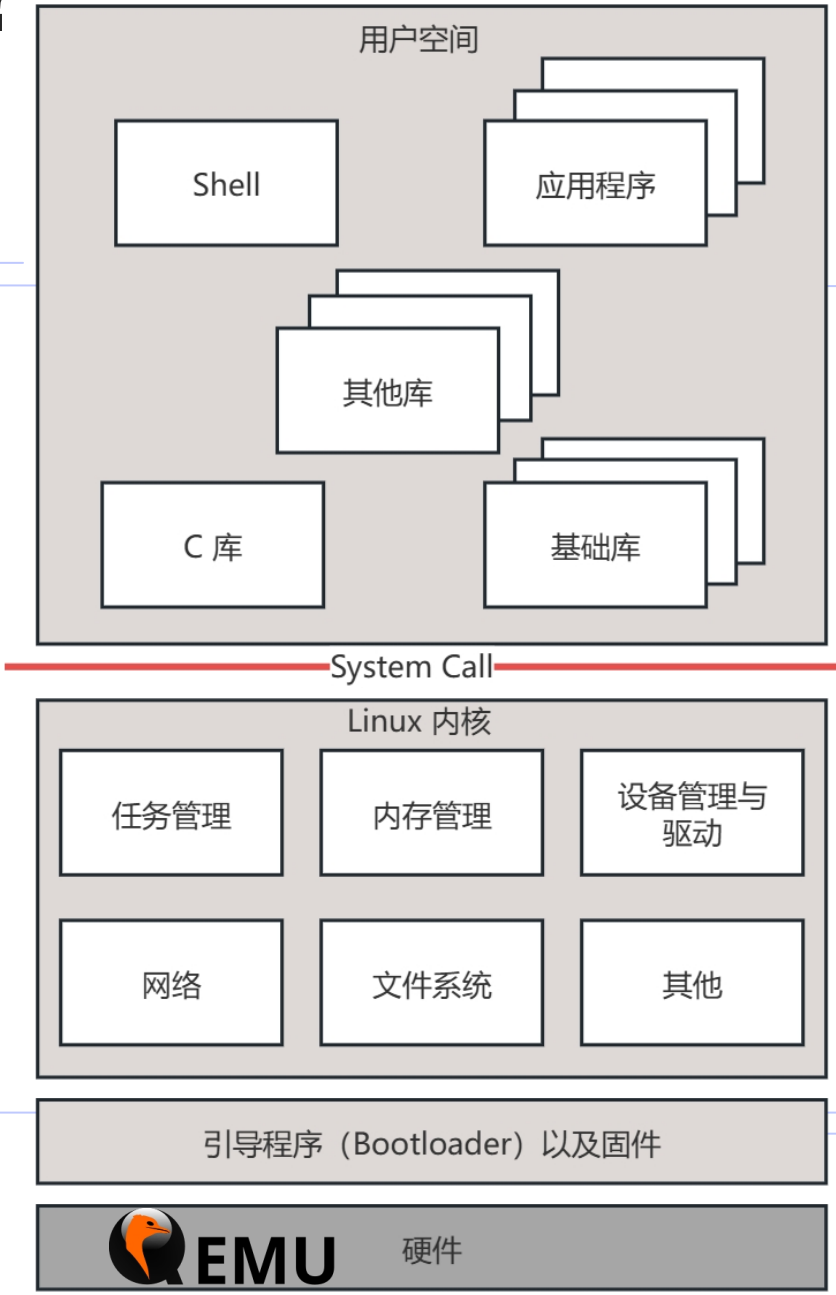


GCC

Binutils

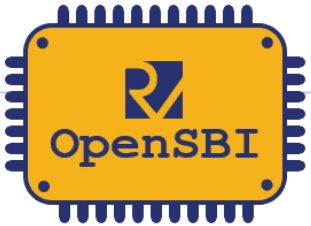
Glibc

需要制作哪些东西？

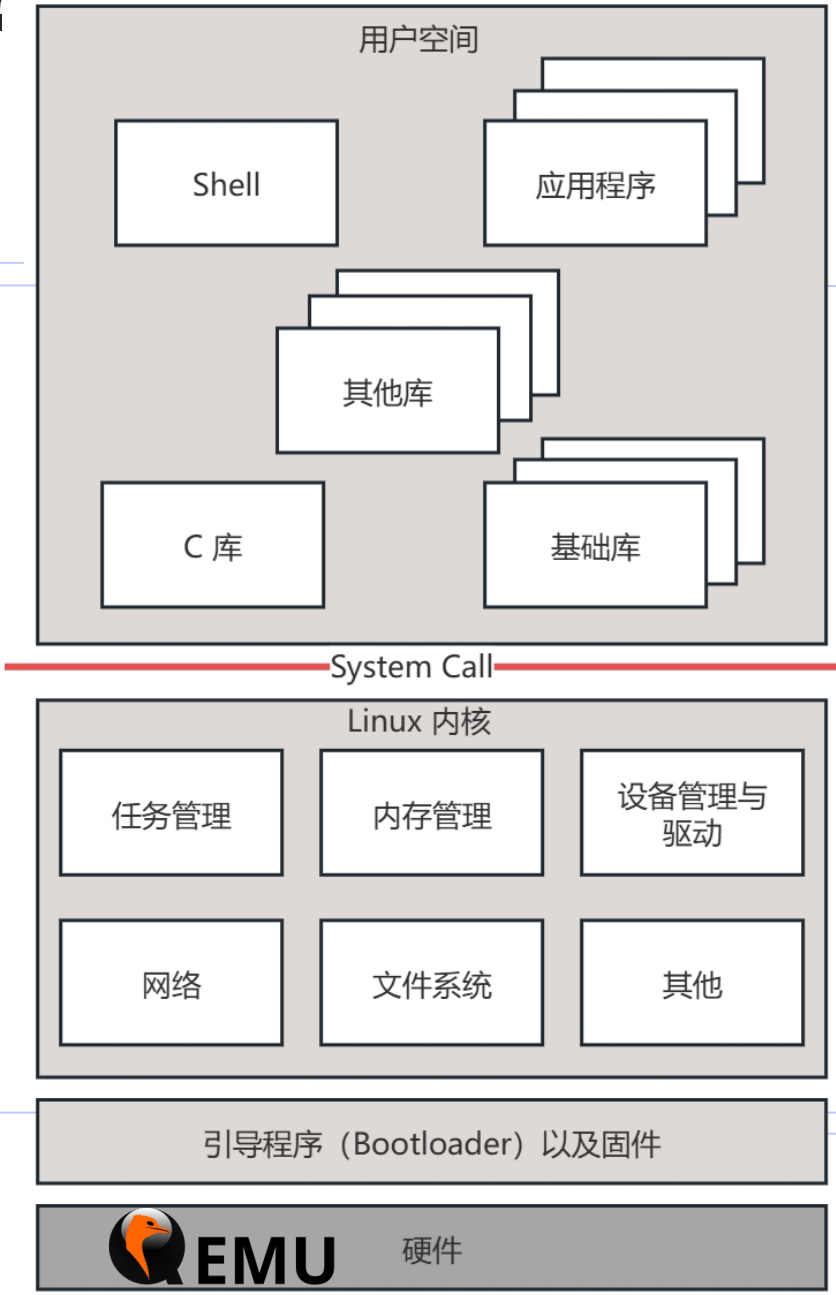


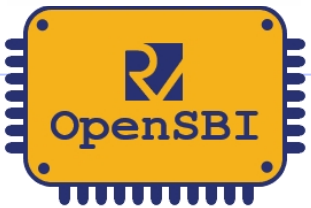
Binutils

Glibc



需要制作哪些东西？



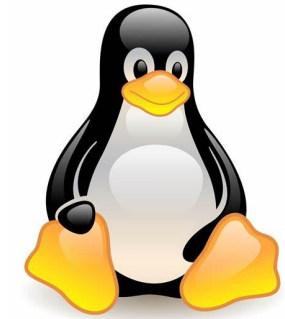


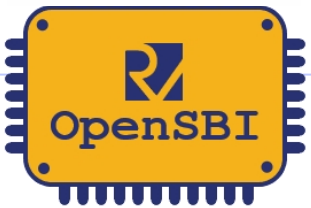
GCC

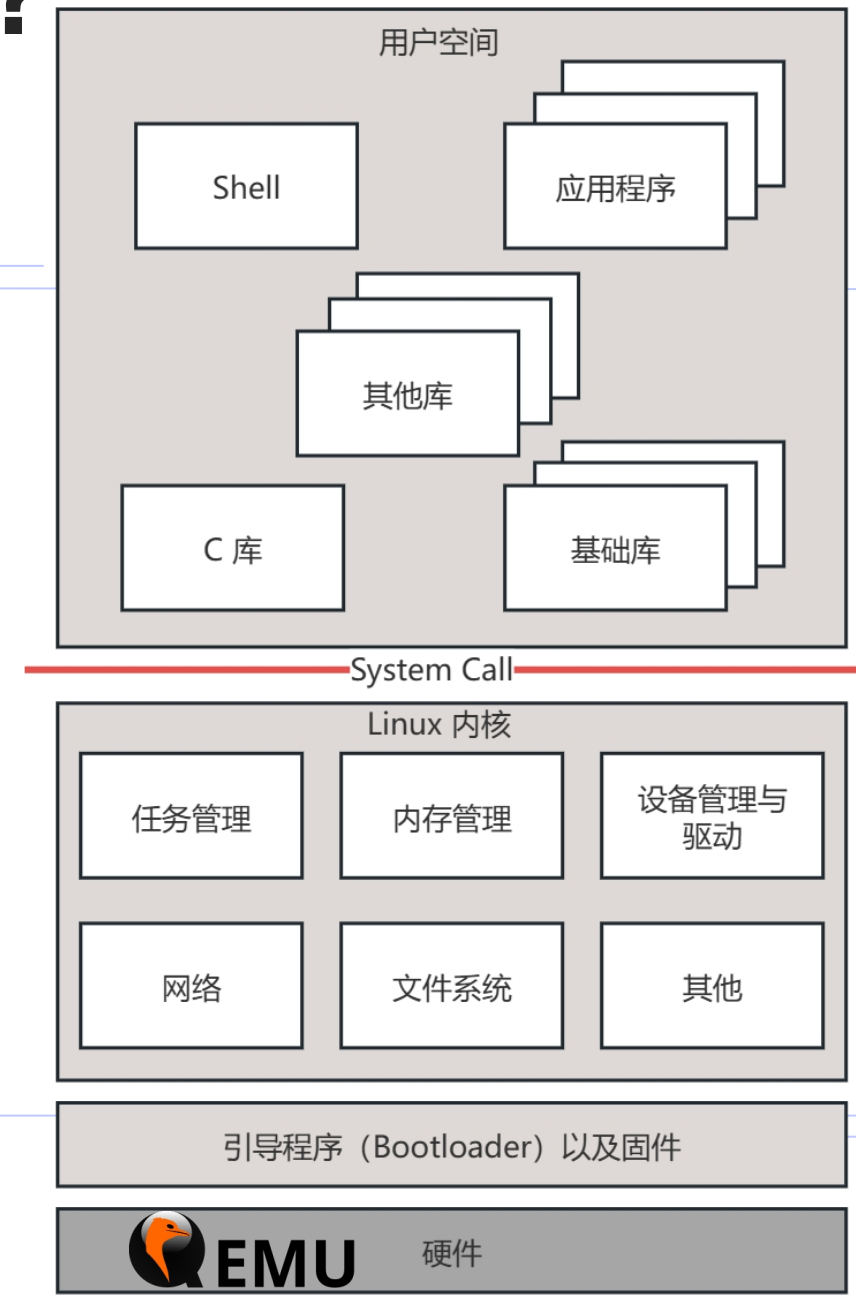
Binutils

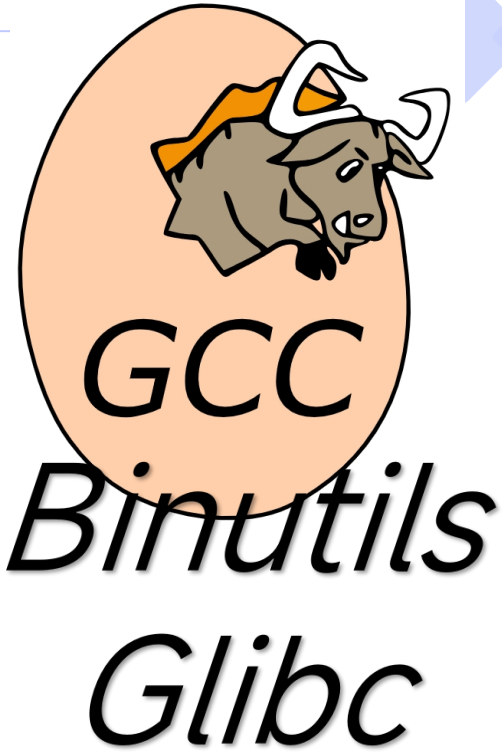
Glibc

需要制作哪些东西？









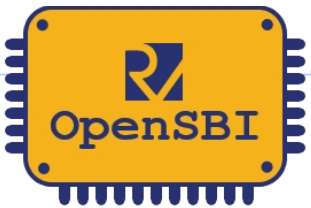
Binutils

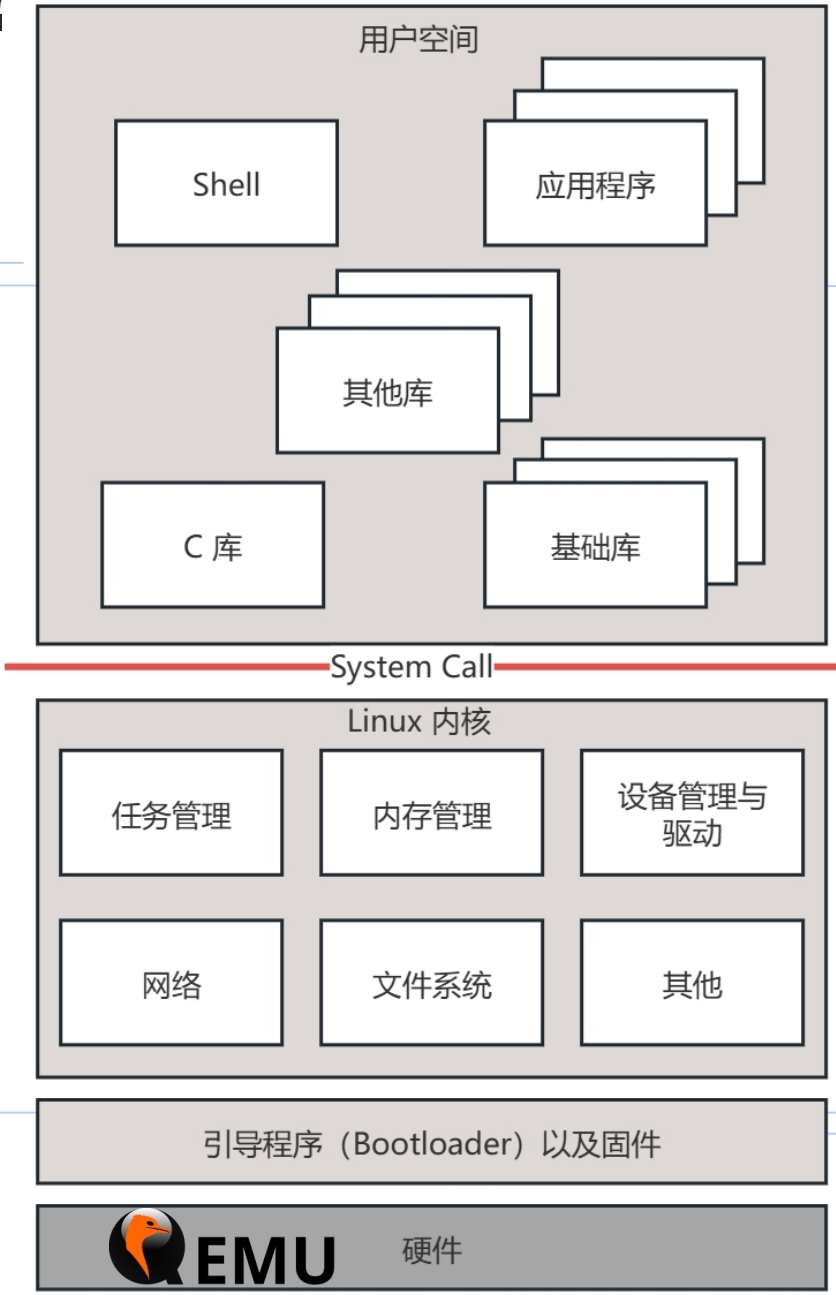
Glibc

需要制作哪些东西？

Coreutils

SysVinit





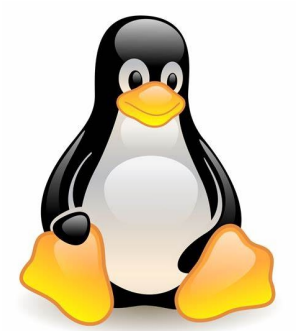
Binutils

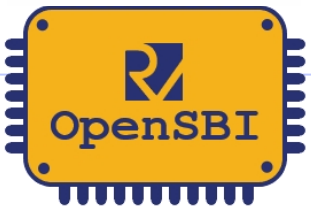
Glibc

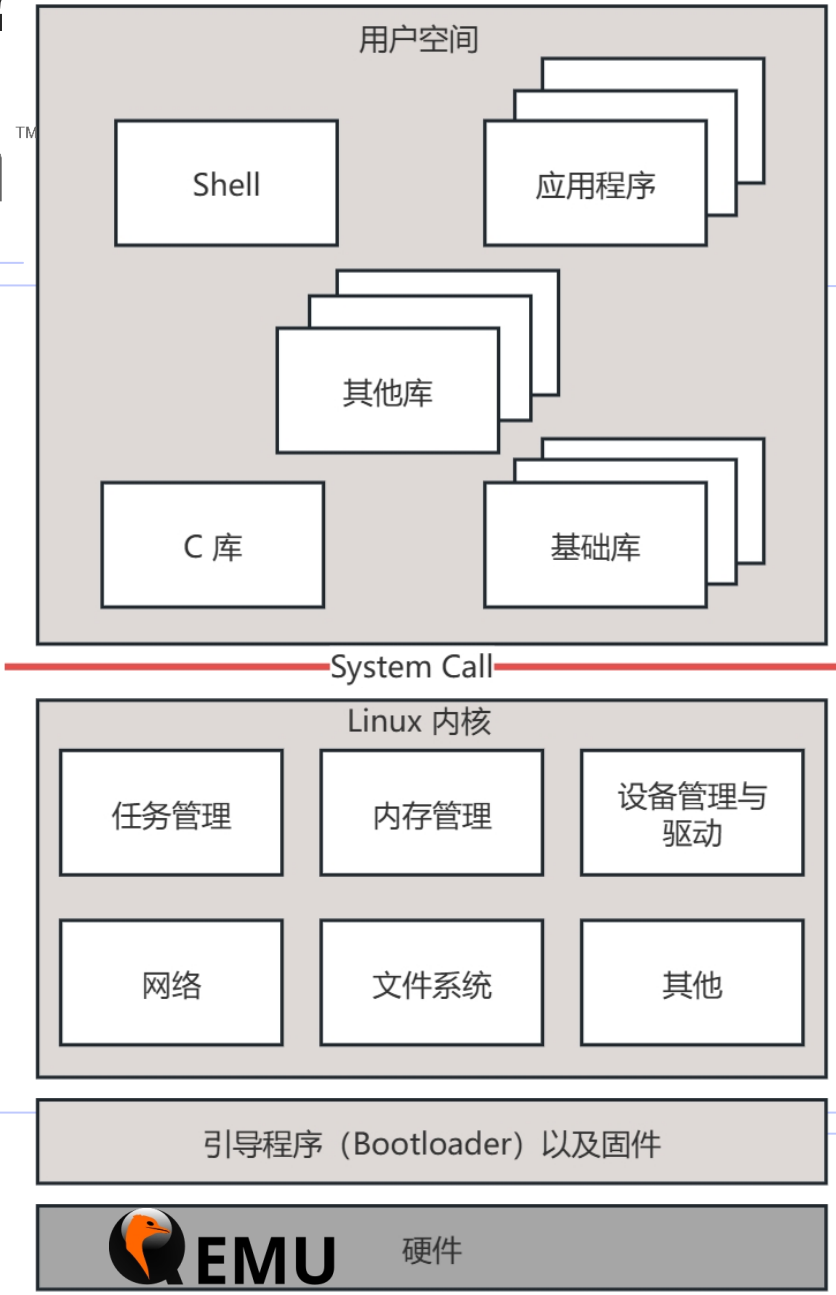
需要制作哪些东西？

Coreutils

SysVinit







Binutils

Glibc

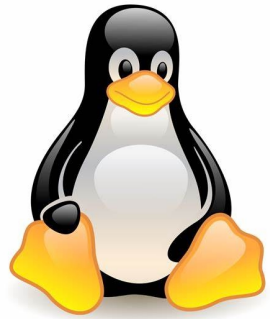
需要制作哪些东西？

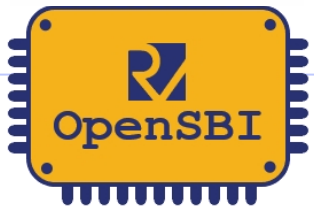
BASH THE BOURNE-AGAIN SHELL

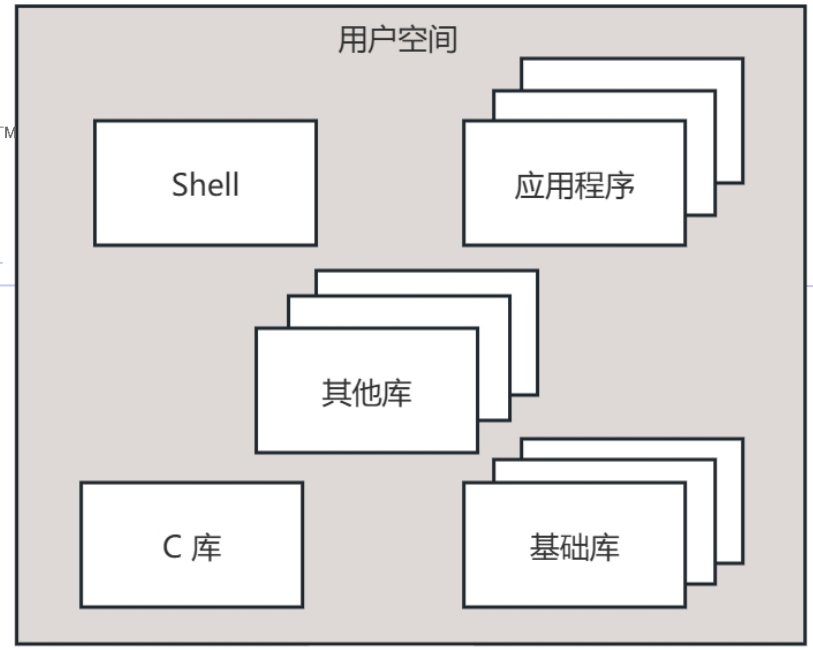
Coreutils

SysVinit

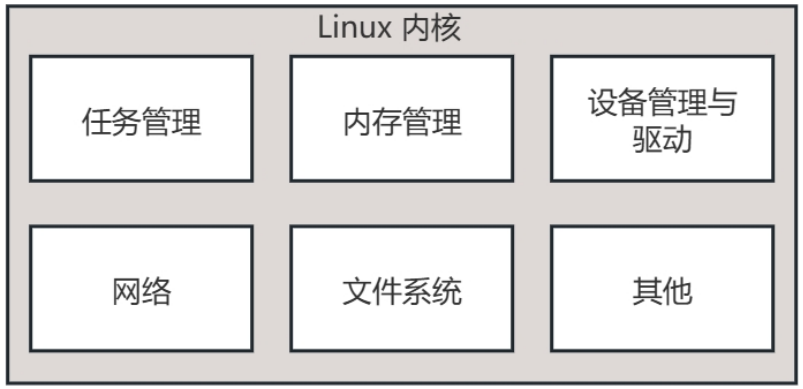
.....

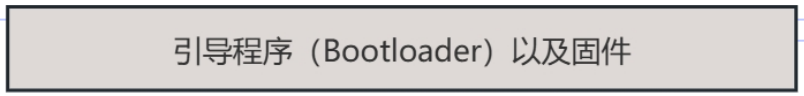






System Call





Binutils

Glibc

02

构建环境准备

构建机器 - 硬件和系统（1）

X86_64

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 24.04.3 LTS
Release:	24.04
Codename:	noble
```

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 22.04.5 LTS
Release:	22.04
Codename:	jammy
```

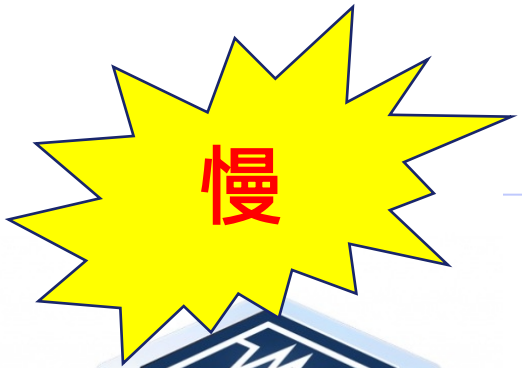
构建机器- 硬件和系统（2）

X86_64

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 24.04.3 LTS
Release:	24.04
Codename:	noble
```

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 22.04.5 LTS
Release:	22.04
Codename:	jammy
```

构建机器- 硬件和系统（2）



X86_64

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:     Ubuntu
Description:  Ubuntu 24.04.3 LTS
Release:      24.04
Codename:   noble
```

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:     Ubuntu
Description:  Ubuntu 22.04.5 LTS
Release:      22.04
Codename:   jammy
```

构建机器 - 预先安装的最小软件工具集合（1）

X86_64

软件名称	注释
which	定位一个命令行程序在文件系统中的位置
sed	stream editor，常用于替换文本内容
make	版本 >= 3.81，通过 build-essential 软件包安装
binutils	提供了 as、ld、objdump、objcopy、readelf 等常用工具，通过同名 binutils 软件包安装
diffutils	用于比较文件差异，包括 diff，cmp 等命令
gcc	版本 >= 4.8，通过 build-essential 软件包安装
g++	版本 >= 4.8，通过 build-essential 软件包安装
bash	著名的 Shell
patch	打补丁用
gzip	数据压缩软件，支持 .gz 格式
bzip2	数据压缩软件，支持 .bz2 格式

构建机器 - 预先安装的最小软件工具集合（2）

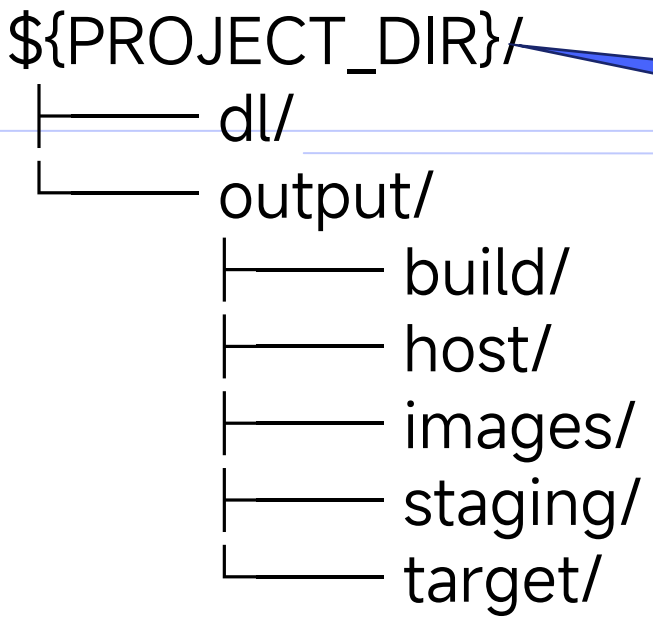
X86_64

软件名称	注释
perl	脚本语言，版本 >= 5.8.7
tar	归档打包软件，和 gzip，bzip2 等结合使用实现打包+压缩
cpio	归档打包软件，和 tar 类似，常用于制作内存盘
unzip	解压 .zip 文件
rsync	文件同步工具
file	必须位于 /usr/bin/file
bc	用于内核配置过程中的数学计算
findutils	提供在文件系统中查找文件和处理文件的工具，如 find、xargs
awk	文本处理和数据分析工具软件
wget	网络下载文件，支持 HTTP/HTTPS/FTP
qemu	版本 >= 6.2.0 安装命令：sudo apt install qemu-system-misc

03

构建目录安排

构建目录安排

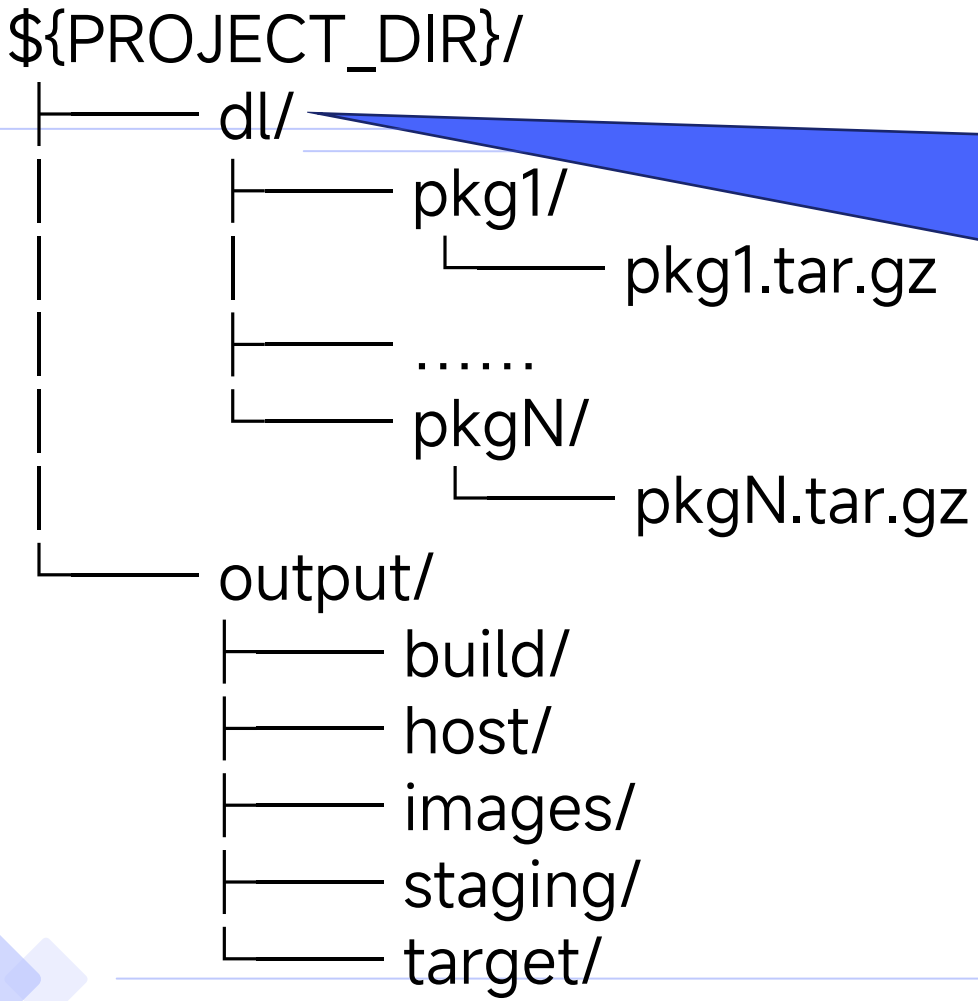


- **${PROJECT_DIR}**: 工作目录

构建目录安排

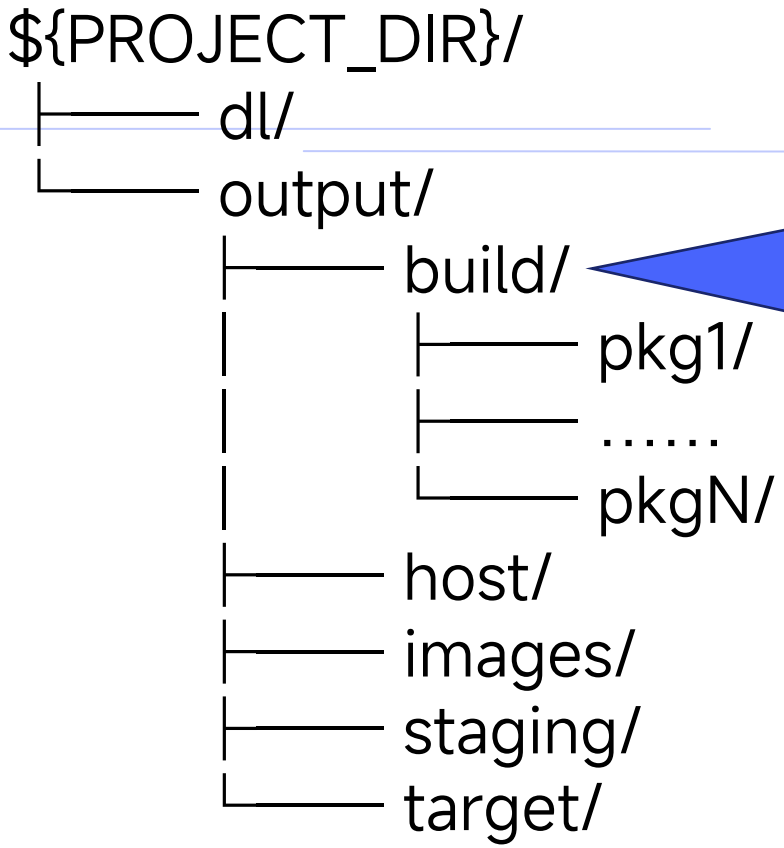
```
${PROJECT_DIR}/
├──── dl/
└──── output/
      ├──── build/
      ├──── host/
      ├──── images/
      ├──── staging/
      └──── target/
```

构建目录安排



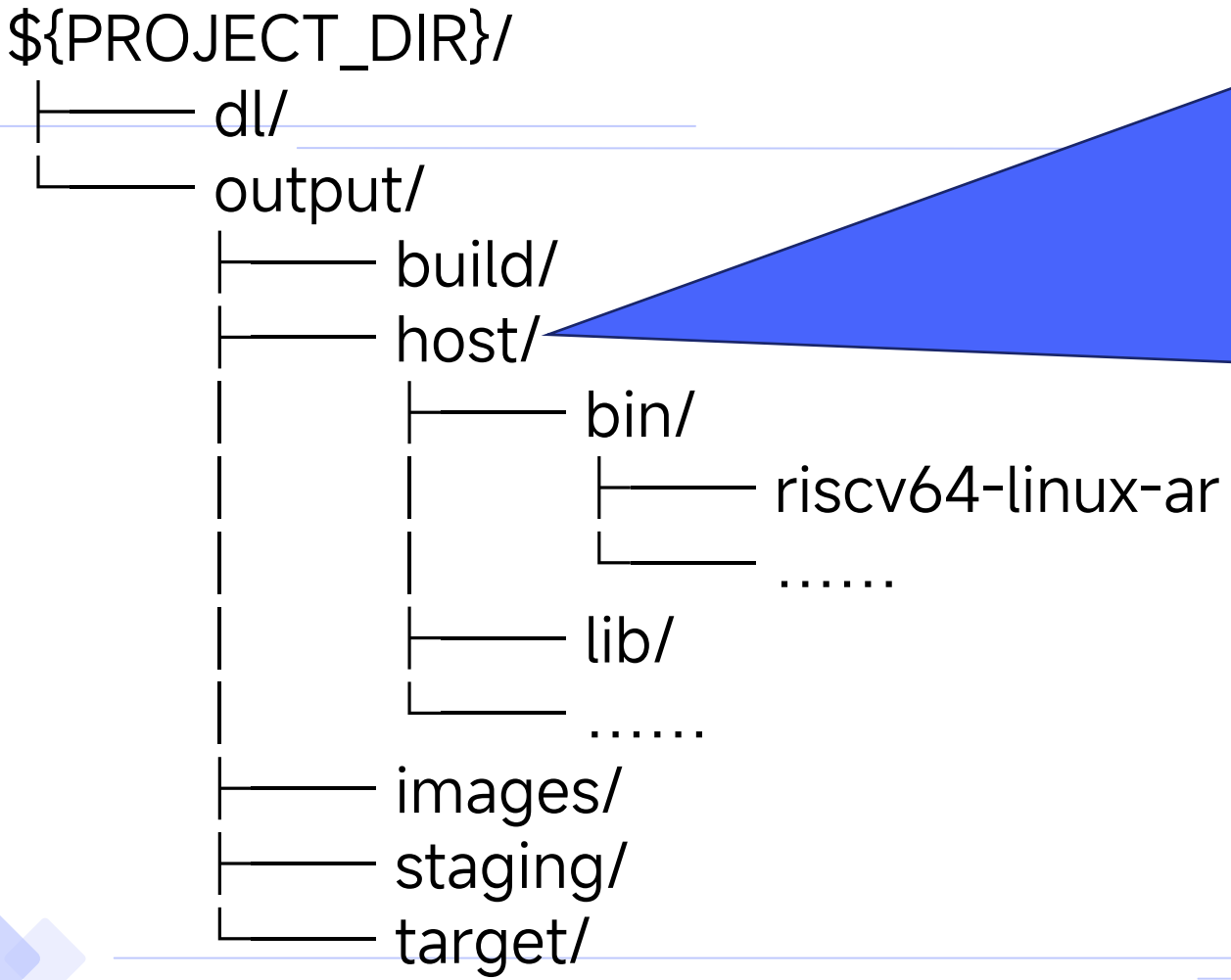
- dl 目录用于存放下载的软件组件的压缩包。每个组件（package）对应一个子目录。
- 对应环境变量 **${DL_DIR}**

构建目录安排



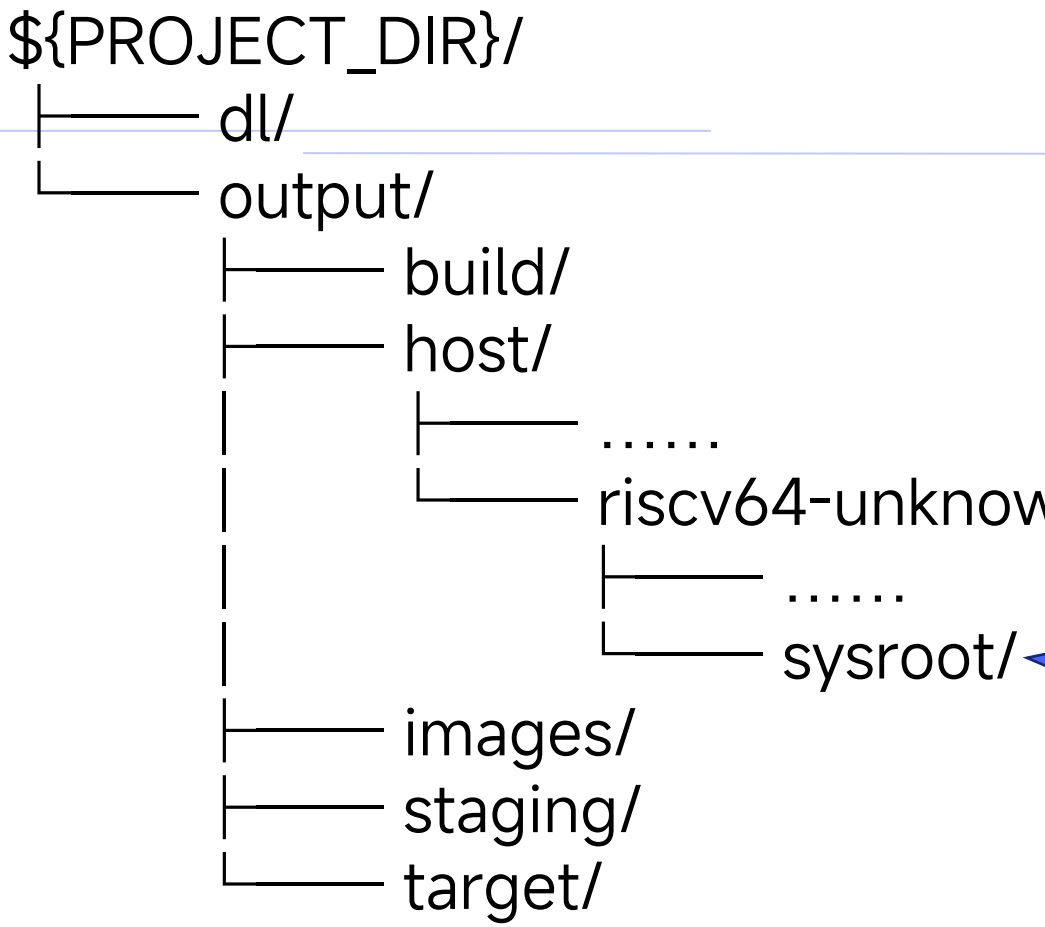
- output/build 目录是构建软件组件的位置。对应环境变量 **${BUILD_DIR}**
- 每个软件组件对应一个子目录。
- 每个软件组件的压缩包将被解压缩到对应的子目录下并在这里完成编译。

构建目录安排



- output/host 目录对应环境变量 **${HOST_DIR}**
- 该目录（不包括 staging 目录）存放了我们自己从源码制作出来的、在本地机器（Host）上运行的可执行程序、库和头文件。
- 特别地，交叉工具链将放在 output/host/bin 下用于制作在目标机器（Target）上运行的软件。

构建目录安排



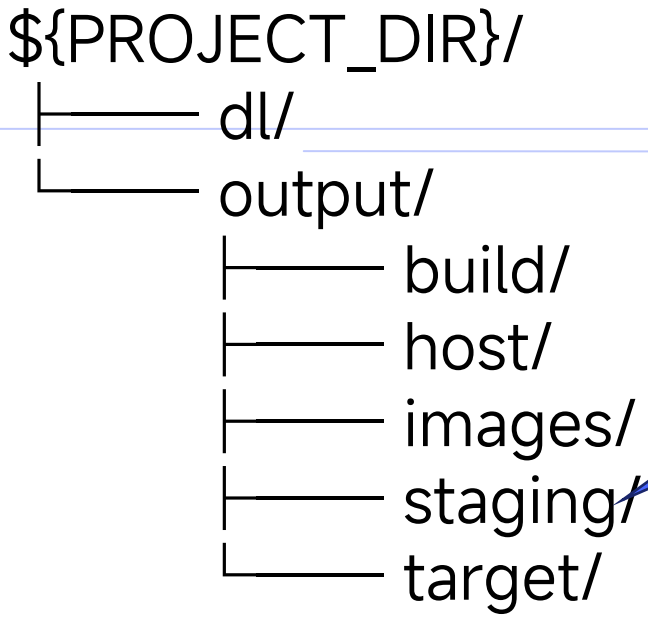
- output/host 目录下有一个特殊的子目录 output/host/riscv64-unknown-linux-gnu/sysroot，叫做暂存（staging）目录。
- 它包含了交叉编译过程中构建软件包所需的头文件（.h，用于编译）、库文件（.so/.a，用于链接）以及其他依赖项。
- Staging 目录和 Target 目录布局保持一致，模拟“虚拟”根文件系统。
- 对应环境变量 **${STAGING_DIR}**

构建目录安排

```
${PROJECT_DIR}/
├── dl/
└── output/
    ├── build/
    ├── host/
    ├── images/
    ├── staging/
    └── target/
```

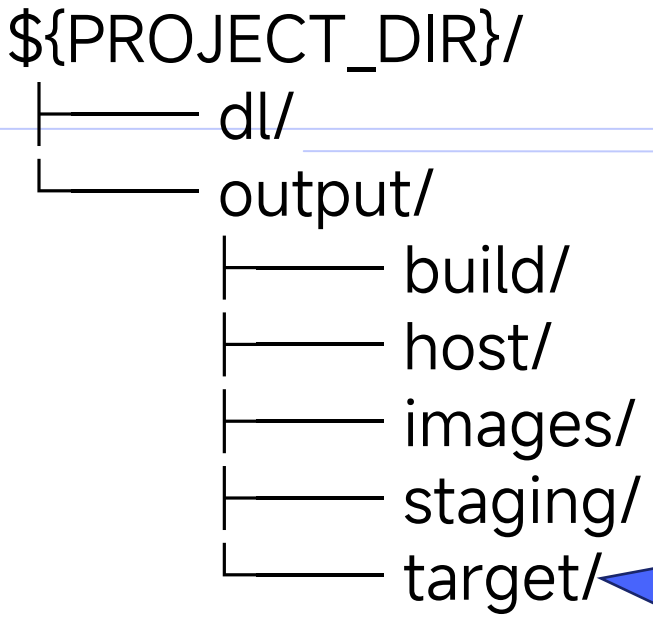
- output/images 目录存放最终生成的内核镜像文件, opensbi 固件和根文件系统镜像文件。
- 对应环境变量 **${IMAGES_DIR}**

构建目录安排



output/stages 是一个符号链接，
指向 **${STAGING_DIR}**

构建目录安排



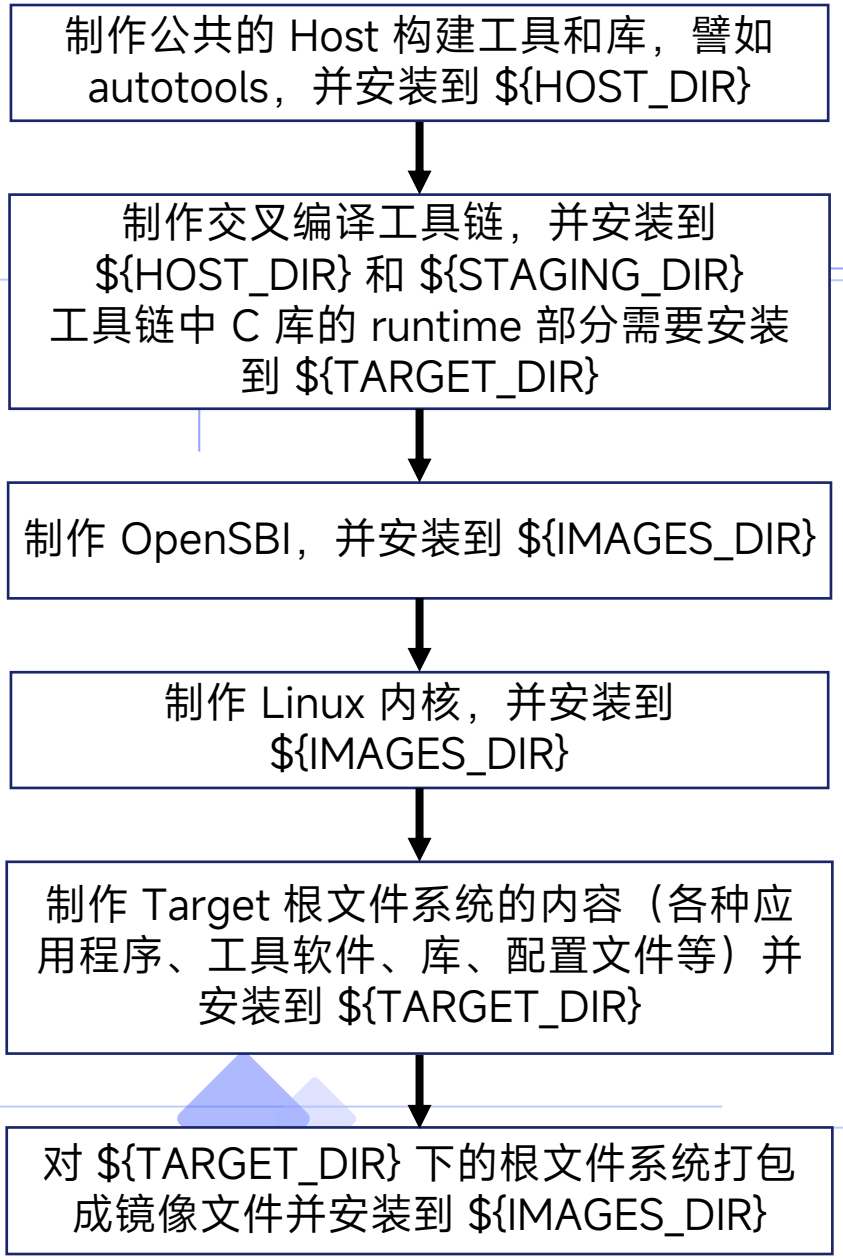
- output/target 按照目标机器（Target） 上根文件系统的目录安排存放我们交叉构建出来的应用程序、共享库（.so）以及其他运行期需要的文件，这个目录下的所有文件最终打包成目标机器的根文件系统镜像用于烧录到目标机器的磁盘上。
- 对应环境变量 **${TARGET_DIR}**

04

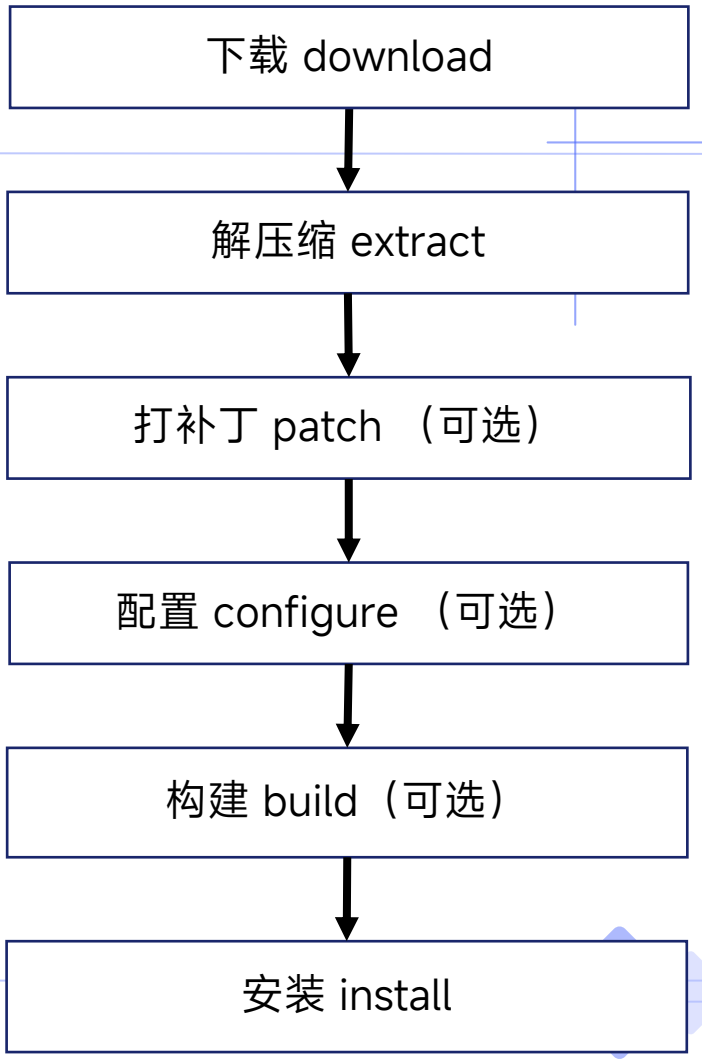
构建流程概览

构建流程概览

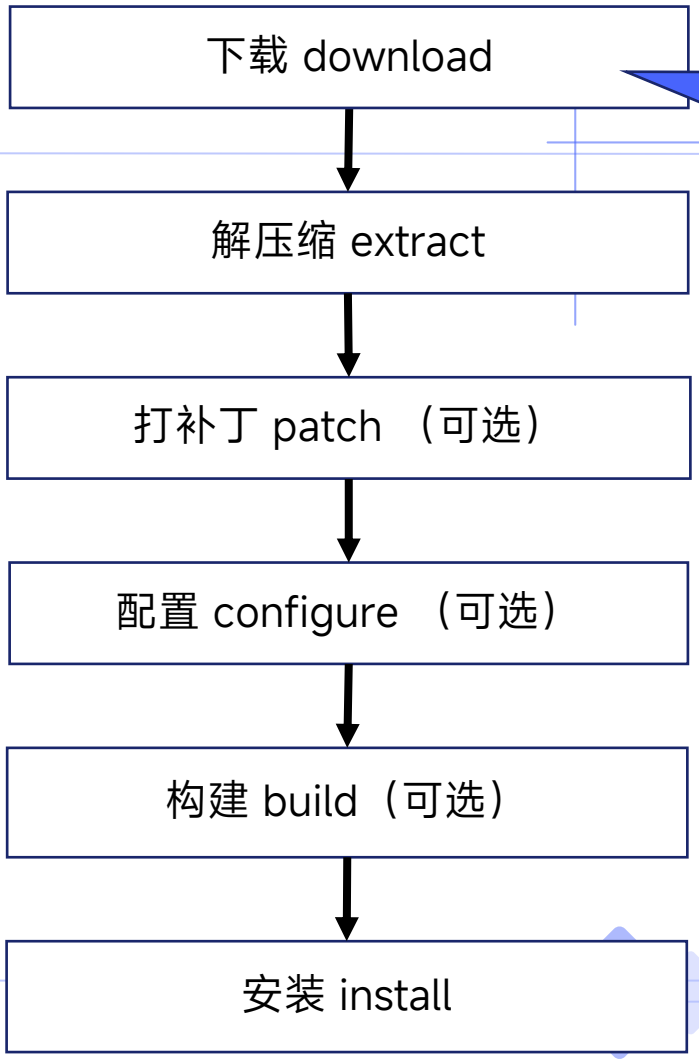
```
${PROJECT_DIR}/
├── dl/
└── output/
    ├── build/
    ├── host/
    ├── images/
    ├── staging/
    └── target/
```



单个软件组件的制作流程



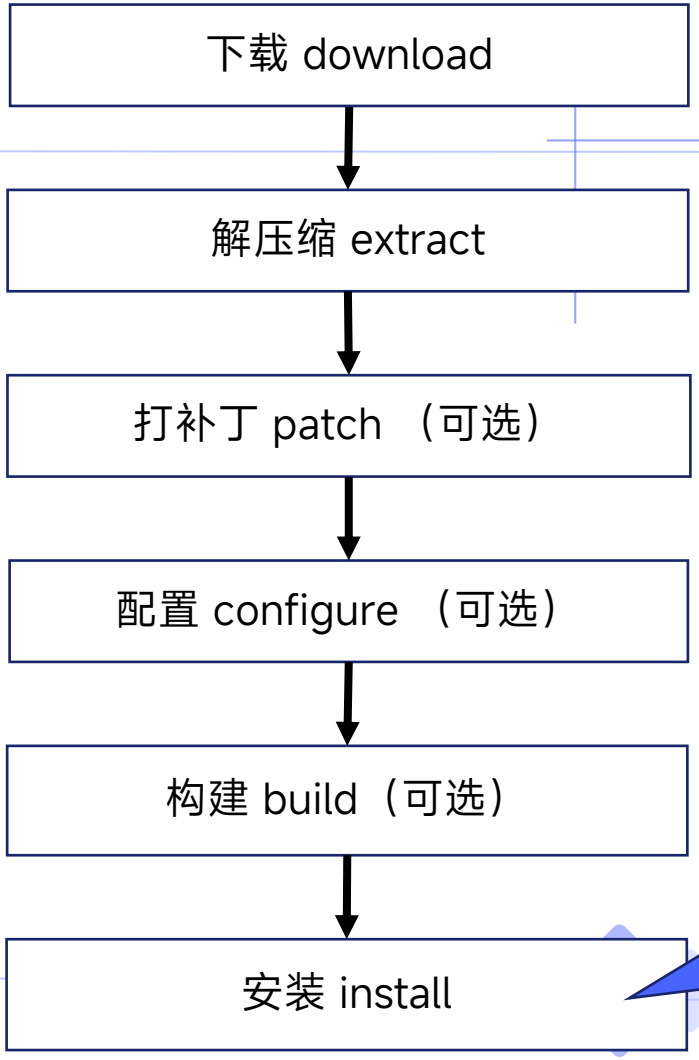
单个软件组件的制作流程



建议在正式构建之前提前将所有软件组件的压缩包都下载下来。

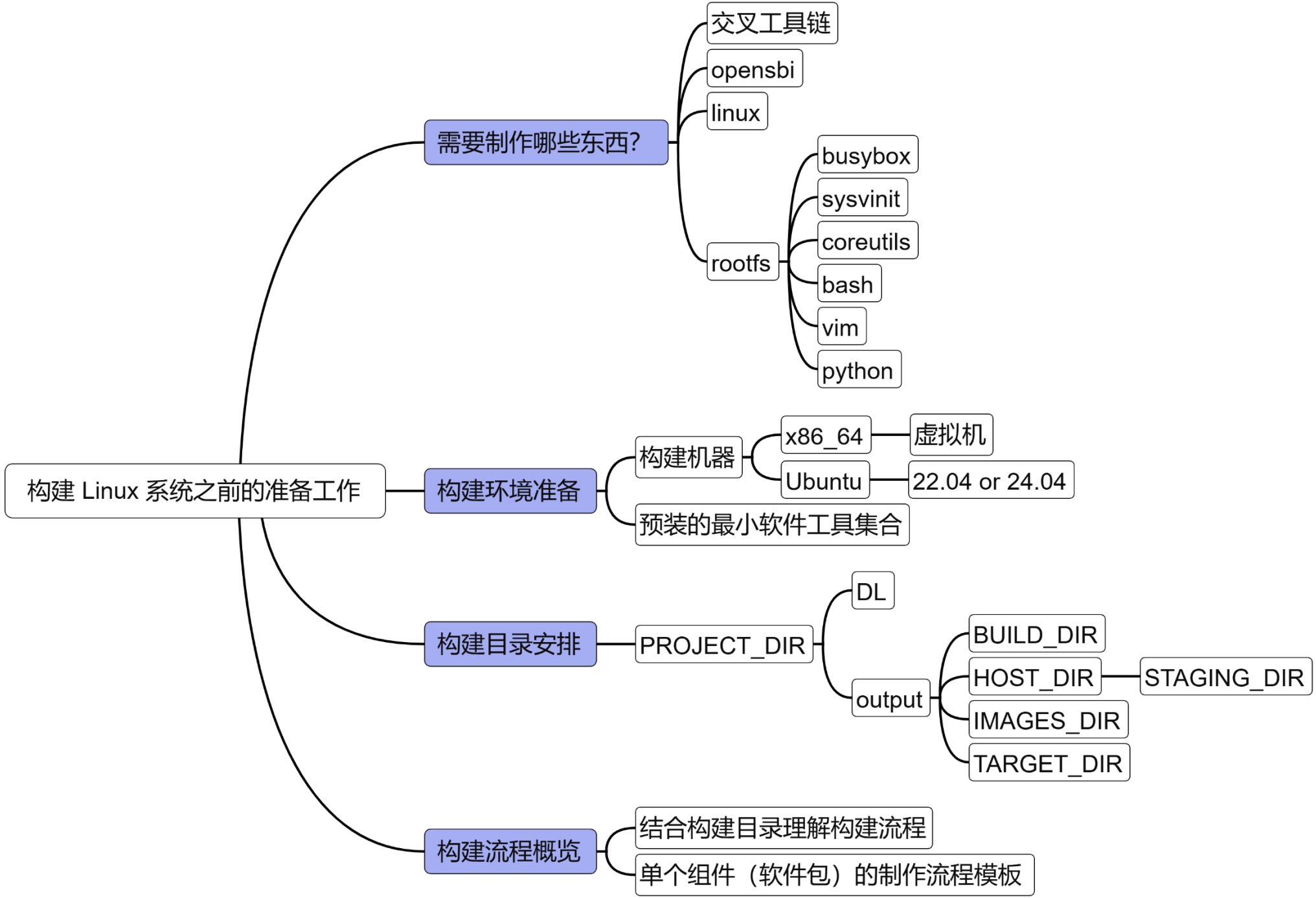
make source

单个软件组件的制作流程



- install-host
- install-staging
- install-target
- install-image

本章总结


构建 Linux 系统之前的准备工作
需要制作哪些东西?
交叉工具链
opensbi
linux
rootfs
busybox
sysvinit
coreutils
bash
vim
python
构建环境准备
构建机器
x86_64
虚拟机
Ubuntu
22.04 or 24.04
预装的最小软件工具集合
构建目录安排
PROJECT_DIR
DL
output
BUILD_DIR
HOST_DIR
STAGING_DIR
IMAGES_DIR
TARGET_DIR
构建流程概览
结合构建目录理解构建流程
单个组件（软件包）的制作流程模板

谢谢