从零开始为 RISC-V 构建一个 Linux 系统

第 8 章 制作文件系统

汪辰

目录

01

基于 Busybox 快速制作一个 Linux 系统

Busybox 简介

BusyBox （https://www.busybox.net/）是一个开源软件，被称为“**嵌入式 Linux 的瑞士军刀**”。

- **高度集成**：将许多常用的 Unix 工具（如 ls, cp, cat, grep, echo, sh 等）集成到一个单一的可执行文件 busybox 中，所有工具都是指向 busybox 的符号链接（symlink）。
- **轻量高效：**体积非常小（通常几百 KB 到几 MB），适合资源受限的环境。代码高度优化，共享通用功能，减少重复。
- **模块化设计**: 编译时可选择需要包含的工具，裁剪不需要的功能。
- **跨平台**: 支持多种硬件架构（ARM、x86、RISC-V 等）；支持多种系统（Linux、Android、FreeBSD 等）。

```
drwxr-xr-x 2 root  root  16384 Jun 23 17:56 ./
drwxr-xr-x 5 64843  300   4096 Jun 23 17:56 ../
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 ash -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 base64 -> busybox*
-rwxr-xr-x 1 root  root 884100 Jun 23 17:56 busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 cat -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 catv -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 chattr -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 chgrp -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 chmod -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 chown -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 conspy -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 cp -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 cpio -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 cttyhack -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 date -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 dd -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 df -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 dmesg -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 dnsdomainname -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 dumpkmap -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 echo -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 ed -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 egrep -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 false -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 fatattr -> busybox*
lrwxrwxrwx 1 root  root      7 Jun 23 17:56 fdflush -> busybox*
```

Busybox – 制作步骤

配置

```
/usr/bin/install -m 0644 -D ${PROJECT_DIR}/package/busybox/busybox.config ${PKGBUILD_DIR}/.config

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"

kconfig_disable_option CONFIG_NOMMU
kconfig_enable_option CONFIG_USE_BB_CRYPT_SHA
kconfig_disable_option CONFIG_PAM
kconfig_enable_option CONFIG_INIT

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"
```

Busybox – 制作步骤

配置

```
/usr/bin/install -m 0644 -D ${PROJECT_DIR}/package/busybox/busybox.config ${PKGBUILD_DIR}/.config

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFI  SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\  AGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/b  AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_CC  -ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFL  \"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y

kconfig_disable_option CONFIG_NOMMU
kconfig_enable_option CONFIG_USE_BB_CRYPT_SH
kconfig_disable_option CONFIG_PAM
kconfig_enable_option CONFIG_INIT

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"
```

先用 busybox.config 替换 .config 文件，busybox.config 是直接参考了 buildroot (2025.08.1) 维护的一套针对 busybox 的默认配置，基于版本 1.36.1。

Busybox - 制作步骤

配置

```
/usr/bin/install -m 0644 -D ${PROJECT_DIR}/package/busybox/busybox.config ${PKGBUILD_DIR}/.config

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"

kconfig_disable_option CONFIG_NOMMU
kconfig_enable_option CONFIG_USE_BB_CRYPT_SHA
kconfig_disable_option CONFIG_PAM
kconfig_enable_option CONFIG_INIT

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LA
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOUF
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CRO
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLA
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"
```

- 基于我们当前的 busybox 版本（1.37.0）升级 .config 文件
- Busybox 只有 oldconfig，没有类似 Linux 的 olddefconfig 命令，而且 Busybox 的 oldconfig 不需要交互确认。

Busybox - 制作步骤

配置

```
/usr/bin/install -m 0644 -D ${PROJECT_DIR}/package/busybox/busybox.config ${PKGBUILD_DIR}/.config

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldcon

kconfig_disable_option CONFIG_NOMMU
kconfig_enable_option CONFIG_USE_BB_
kconfig_disable_option CONFIG_PAM
kconfig_enable_option CONFIG_INIT

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LAR
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOUR
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROS
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLA
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y old
```

手动调整一些 Busybox 的配置:

- 关闭 CONFIG_NOMMU：QEMU riscv64 virt 支持 MMU。
- 使能 CONFIG_USE_BB_CRYPT_SHA：使用 BusyBox 自己的 SHA 加密算法，节省内存但安全性较低。
- 关闭 CONFIG_PAM：使用 BusyBox 内置的简单密码验证
- 使能 CONFIG_INIT：使用 Busybox 内置的简化版本 init 程序。

Busybox - 制作步骤

配置

```
/usr/bin/install -m 0644 -D ${PROJECT_DIR}/package/busybox/busybox.config ${PKGBUILD_DIR}/.config

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\" AR=\"${CROSS_COMPILE}gcc-ar\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\"
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"

kconfig_disable_option CONFIG_NOMMU
kconfig_enable_option CONFIG_USE_BB_CRYPT_SHA
kconfig_disable_option CONFIG_PAM
kconfig_enable_option CONFIG_INIT

yes "" | eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_
D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS
j${MAXNUM_CPUS} -C ${PKGBUILD_DIR} HOSTCC=\"/usr/bin/gcc\"
NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPI
ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\"
CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y oldconfig"
```

调整过配置后再同步刷新一下 .config。

Busybox - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -
D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\"
CFLAGS_busybox=\"\" /usr/bin/make -j${MAXNUM_CPUS} AR=\"${CROSS_COMPILE}gcc-
ar\" NM=\"${CROSS_COMPILE}${CROSS_COMPILE}gcc-nm\"
RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\" ARCH=riscv
PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\"
CROSS_COMPILE=\"${CROSS_COMPILE}\" CONFIG_PREFIX=\"${TARGET_DIR}\"
SKIP_STRIP=y -C ${PKGBUILD_DIR}"
```

Busybox – 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -
D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\"
CFLAGS_busybox=\"\" /usr/bin/make -j${MAXNUM_CPUS} AR=\"${CROSS_COMPILE}gcc-
ar\" NM=\"${CROSS_COMPILE}${CROSS_COMPILE}gcc-nm\"
RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\" ARCH=riscv
PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\"
CROSS_COMPILE=\"${CROSS_COMPILE}\" CONFIG_PREFIX=\"${TARGET_DIR}\"
SKIP_STRIP=y -C ${PKGBUILD_DIR}"
```

指定 BusyBox 二进制文件和符号链接在本地主机上的安装目录前缀，默认会安装
到 $(PREFIX)/bin/ 目录下。

Busybox – 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -
D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\"
CFLAGS_busybox=\"\" /usr/bin/make -j${MAXNUM_CPUS} AR=\"${CROSS_COMPILE}gcc-
ar\" NM=\"${CROSS_COMPILE}${CROSS_COMPILE}gcc-nm\"
RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\" ARCH=riscv
PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\"
CROSS_COMPILE=\"${CROSS_COMPILE}\" CONFIG_PREFIX=\"${TARGET_DIR}\"
SKIP_STRIP=y -C ${PKGBUILD_DIR}"
```

指定 BusyBox 二进制文件和符号链接在目标主机上的安装目录前缀。

Busybox - 制作步骤

安装 （install-target）

```
eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -j${MAXNUM_CPUS} AR=\"${CROSS_COMPILE}gcc-ar\" NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\" ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\" CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y -C ${PKGBUILD_DIR} install-noclobber"
```

Busybox - 制作步骤

安装 （install-target）

```
eval "${TARGET_MAKE_ENV} CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -D_FORTIFY_SOURCE=1\" CFLAGS_busybox=\"\" /usr/bin/make -j${MAXNUM_CPUS} AR=\"${CROSS_COMPILE}gcc-ar\" NM=\"${CROSS_COMPILE}gcc-nm\" RANLIB=\"${CROSS_COMPILE}gcc-ranlib\" CC=\"${CROSS_COMPILE}gcc\" ARCH=riscv PREFIX=\"${TARGET_DIR}\" EXTRA_LDFLAGS=\"\" CROSS_COMPILE=\"${CROSS_COMPILE}\" CONFIG_PREFIX=\"${TARGET_DIR}\" SKIP_STRIP=y -C ${PKGBUILD_DIR} install-noclobber"
```

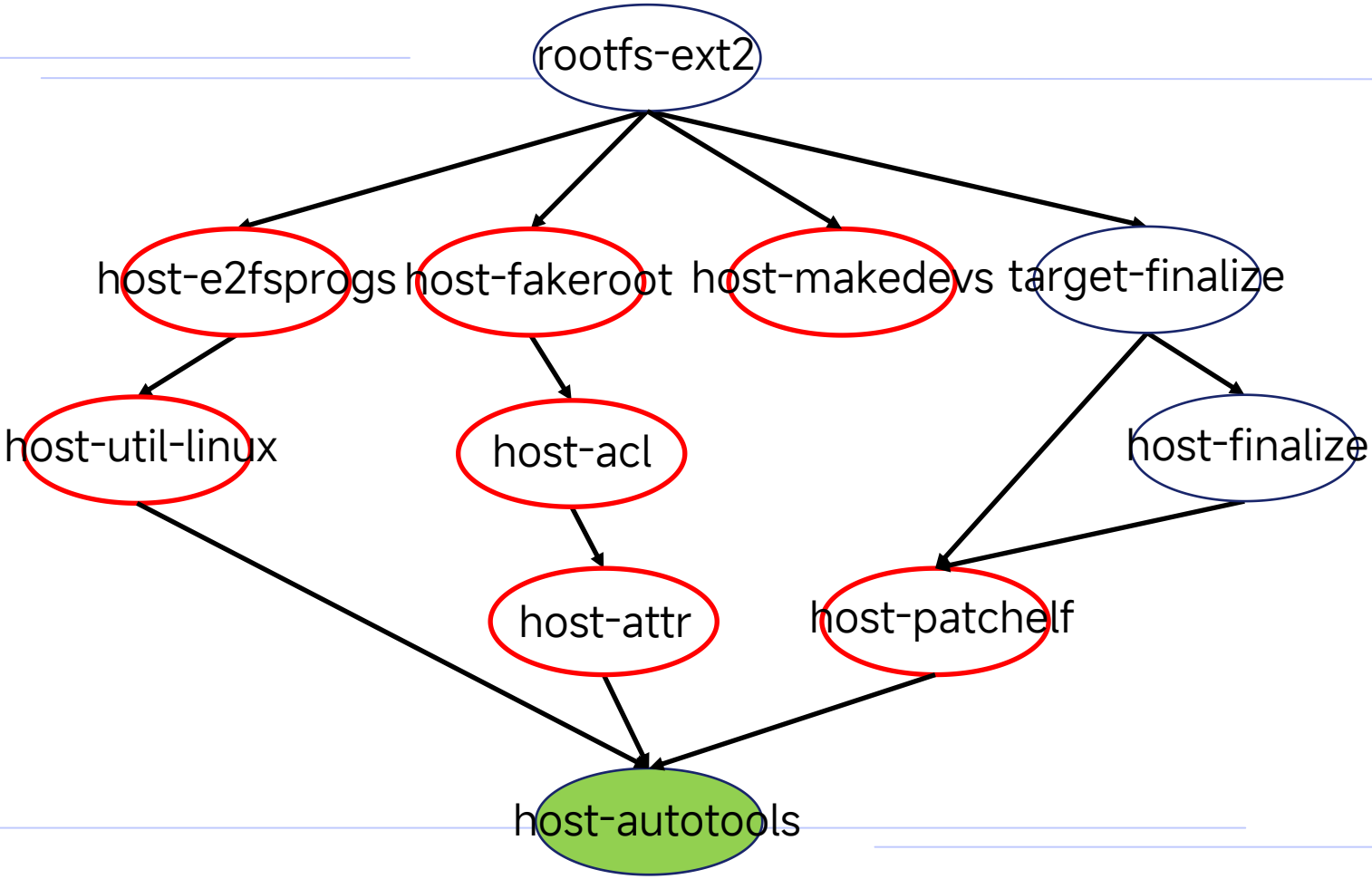
和 make install 不同，make install-noclobber 会跳过已存在的文件

Busybox – 制作步骤

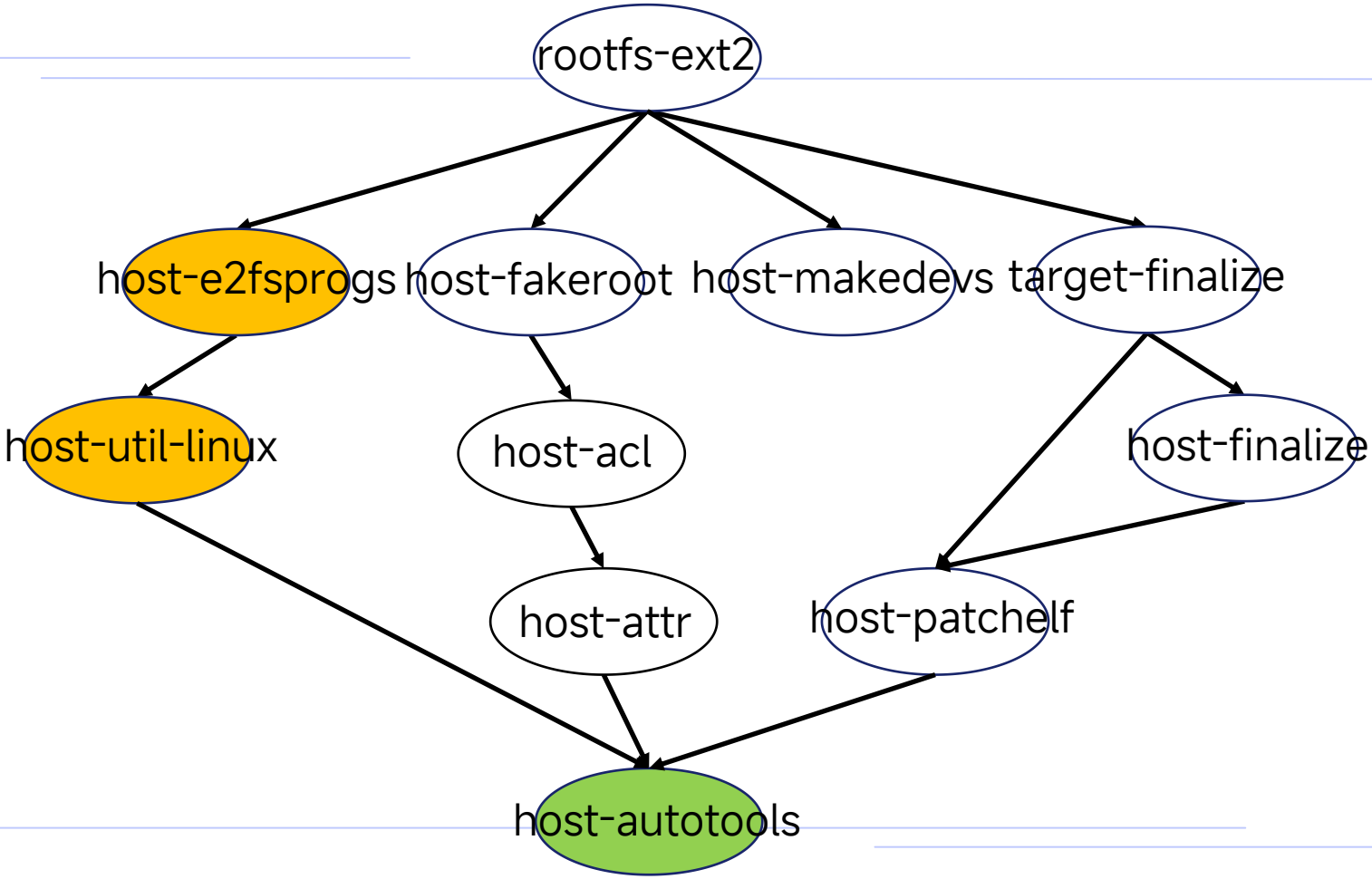
安装 （install-target）

```
if test ! -e ${TARGET_DIR}/etc/inittab; then
        /usr/bin/install -D -m 0644 ${PROJECT_DIR}/package/busybox/inittab ${TARGET_DIR}/etc/inittab;
fi
if grep -q CONFIG_UDHCPC=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/udhcpc.script ${TARGET_DIR}/usr/share/udhcpc/default.script;
        /usr/bin/install -m 0755 -d ${TARGET_DIR}/usr/share/udhcpc/default.script.d;
fi
if grep -q CONFIG_ZCIP=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PKGBUILD_DIR}/examples/zcip.script ${TARGET_DIR}/usr/share/zcip/default.script;
fi
if grep -q CONFIG_SYSLOGD=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S01syslogd ${TARGET_DIR}/etc/init.d/S01syslogd;
fi;
if grep -q CONFIG_KLOGD=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S02klogd ${TARGET_DIR}/etc/init.d/S02klogd;
fi
if grep -q CONFIG_BB_SYSCTL=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S02sysctl ${TARGET_DIR}/etc/init.d/S02sysctl ;
fi
if grep -q CONFIG_IFPLUGD=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S41ifplugd ${TARGET_DIR}/etc/init.d/S41ifplugd;
fi;
if grep -q CONFIG_CROND=y ${PKGBUILD_DIR}/.config; then
        mkdir -p ${TARGET_DIR}/etc/cron/crontabs
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S50crond ${TARGET_DIR}/etc/init.d/S50crond
fi;
if grep -q CONFIG_FEATURE_TELNETD_STANDALONE=y ${PKGBUILD_DIR}/.config; then
        /usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/busybox/S50telnet ${TARGET_DIR}/etc/init.d/S50telnet ;
fi
```

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

Util-linux 简介

https://github.com/util-linux/util-linux

- util-linux 就像一个“系统管理员的瑞士军刀”，提供了管理 Linux 系统所需的各种底层工具以及关键的共享库，是维持系统正常运转的基础组件。包含的工具涵盖以下几大类：
  - ✓ **磁盘与存储管理：**负责分区、格式化、挂载、查看块设备信息，如：fdisk, mount, mkfs, lsblk, findfs
  - ✓ **系统信息与调试：**负责查看内核信息、硬件详细信息，如：dmesg, lscpu, lsmem, fdisk
  - ✓ **用户与登录管理：**负责用户登录、会话管理，如：login, logout
  - ✓ **其他系统工具：**负责文本处理、任务调度、定时关机等，如 column, flock, wall
- 我们这里需要先构建 util-linux，因为它是构建 e2fsprogs 的依赖。

Util-linux – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking  --without-systemd --with-systemdsystemunitdir=no --without-python --enable-libblkid --enable-libuuid --enable-libmount --without-libmagic --without-ncurses --without-ncursesw --without-tinfo --disable-raw --disable-makeinstall-chown --disable-agetty --disable-chfn-chsh --disable-chmem --disable-ipcmk --disable-liblastlog2 --disable-login --disable-lsfd --disable-lslogins --disable-mesg --disable-more --disable-newgrp --disable-nologin --disable-nsenter --disable-pg --disable-rfkill --disable-runuser --disable-schedutils --disable-setpriv --disable-setterm --disable-su --disable-sulogin --disable-tunelp --disable-ul --disable-unshare --disable-uuidd --disable-vipw --disable-wall --disable-wdctl --disable-write --disable-zramctl"
```

Util-linux – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --
localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-
debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking  --
without-systemd --with-systemdsystemunitdir=no --without-python --enable-libblkid --
enable-libuuid --enable-libmount --without-libmagic --without-ncurses --without-
ncursesw --without-tinfo --disable-raw --disable-makeinstall-chown --disable-agetty --
disable-chfn-chsh --disable-chmem --disable-ipcmk --disable-logger --disable-login --
disable-lsfd --disable-lslogins --disable-mesg --disable-newgrp --disable-
nologin --disable-nsenter --disable-pg --disable-rfkill --disable-
schedutils --disable-setpriv --disable-setterm --disable-su --disable-
tunelp --disable-ul --disable-unshare --disable-uuidd --disable-wall --
disable-wdctl --disable-write --disable-zramctl
```

e2fsprogs 的构建明确依赖于 libblkid（用于块设备识别）和 libuuid（用于生成全局唯一标识符 UUID）这两个库。

Util-linux – 制作步骤

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

e2fsprogs 简介

http://e2fsprogs.sourceforge.net

- e2fsprogs（Ext2/3/4 Filesystem Programs）是一套用于创建、维护、检查和调试 ext2、ext3 和 ext4 文件系统的核心工具集。这些文件系统是绝大多数 Linux 发行版的默认或历史标准。 包含的工具涵盖以下几大类：
  - ✓ **创建工具**：格式化分区，创建文件系统。 如： mkfs.ext2/ext3/ext4
  - ✓ **检查修复工具**：检查并修复文件系统错误，，如： e2fsck
  - ✓ **属性调整工具**：查看或调整文件系统参数（如标签、UUID、检查间隔）。 如：tune2fs
  - ✓ **调试高级工具**：提供交互式文件系统调试，可手动修复严重问题，如 debugfs
  - ✓ **其他实用工具**：转储超级块信息、在线调整大小、查看文件碎片情况等，如 dumpe2fs, resize2fs, filefrag
- 我们这里制作 rootfs 镜像时采用 EXT2 文件格式，需要用到 mkfs.ext2。

e2fsprogs - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_path_LDCONFIG=true
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --
disable-dependency-tracking  --disable-defrag --disable-e2initrd-helper --disable-fuse2fs
--disable-fsck --disable-libblkid --disable-libuuid --disable-testio-debug --enable-symlink-
install --enable-elf-shlibs --with-crond-dir=no --with-udev-rules-dir=no --with-systemd-
unit-dir=no"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR} install install-libs"

rm -f ${HOST_DIR}/bin/compile_et
```

e2fsprogs - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_path_LDCONFIG=true
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --
disable-dependency-tracking  --disable-defrag --disable-e2initrd-helper --disable-fuse2fs
--disable-fsck --disable-libblkid --disable-libuuid --disable-testio-debug --enable-symlink-
install --enable-elf-shlibs --with-c    nd-dir=no --with-udev-rules-dir=no --with-systemd-
unit-dir=no"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/                                                    BUILD_DIR}"
```

明确选择不使用 e2fsprogs 自带的 libblkid 和 libuuid，而是使用 util-linux 提供的 libblkid 和 libuuid。

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR} install install-libs"

rm -f ${HOST_DIR}/bin/compile_et
```

e2fsprogs - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_path_LDCONFIG=true
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --
disable-dependency-tracking  --disable-defrag --disable-e2initrd-helper --disable-fuse2fs
--disable-fsck --disable-libblkid --disable-libuuid --disable-testio-debug --enable-symlink-
install --enable-elf-shlibs --with-crond-dir=no --with-udev-rules-dir=no --with-systemd-
unit-dir=no"
```

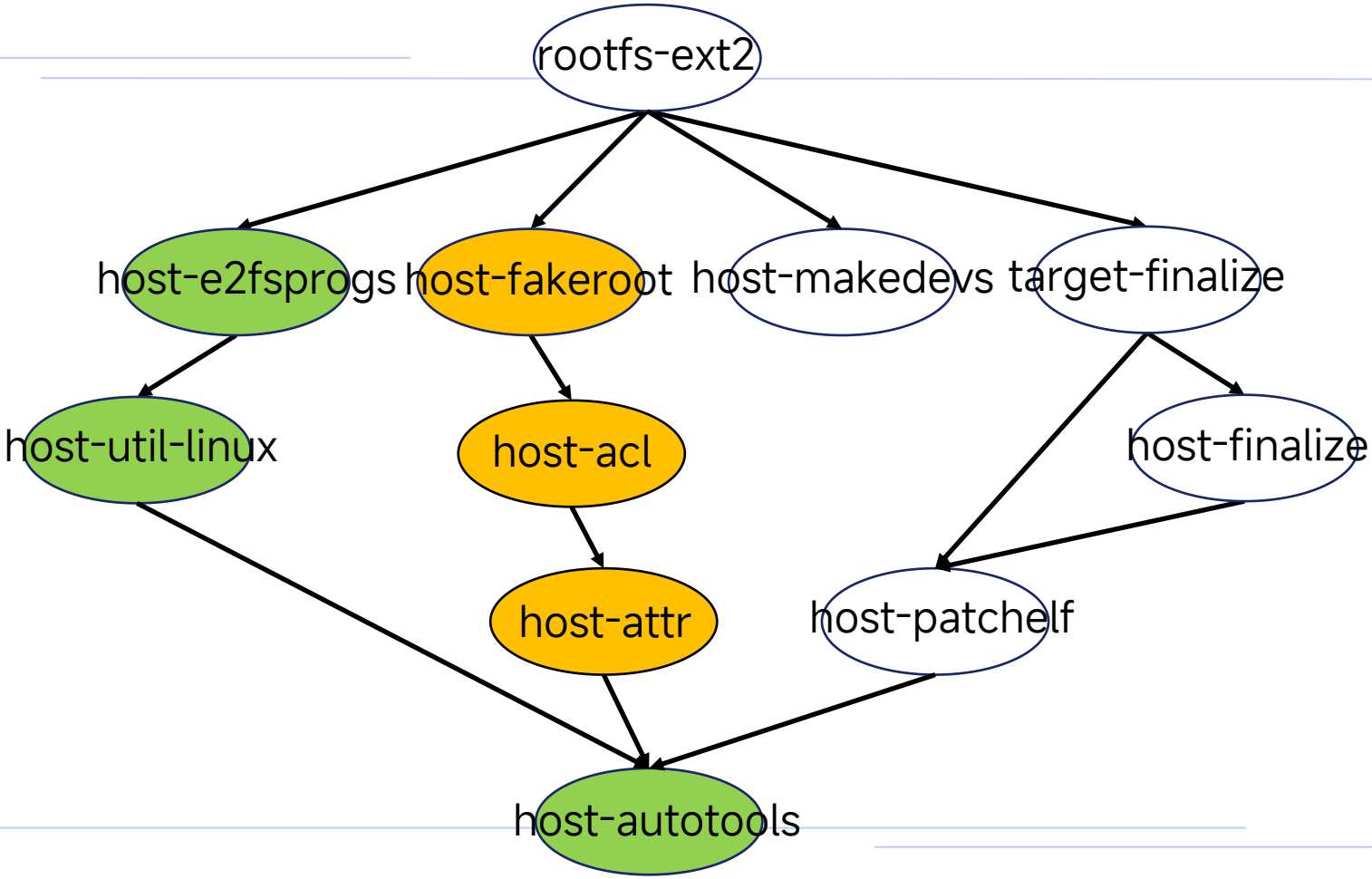
构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR} install install-libs"

rm -f ${HOST_DIR}/bin/compile_et
```

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

attr 简介

https://savannah.nongnu.org/projects/attr

- attr 是一个用于管理 Linux 文件系统上的扩展属性（Extended Attributes，简称 xattr）的软件包。
- attr 软件包提供了一系列工具 attr，getfattr 和 setfattr 以及 libattr 库
- attr 是一套让你能深入利用现代 Linux 文件系统高级元数据功能的工具。如支持 SELinux 环境、实现精细的文件管理或备份等。
- 构建 acl 软件包时依赖于 attr 的 libattr 库和相关头文件。

attr - 制作步骤

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i“

patch_libtool ${PKGBUILD_DIR}

eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --
localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --
disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-
debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

acl 简介

http://savannah.nongnu.org/projects/acl

- acl 是一个提供访问控制列表（ Access Control List ）功能的软件包，它实现了比传统 Unix 权限更精细的 Linux 文件权限管理机制。
- acl 软件包提供了一系列工具 getfacl，setfacl 以及 libacl 库
- 构建 acl 时依赖 attr，因为 acl 需要调用 attr 提供的库来访问和操作文件的扩展属性

acl - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

fakeroot 简介

https://wiki.debian.org/FakeRoot

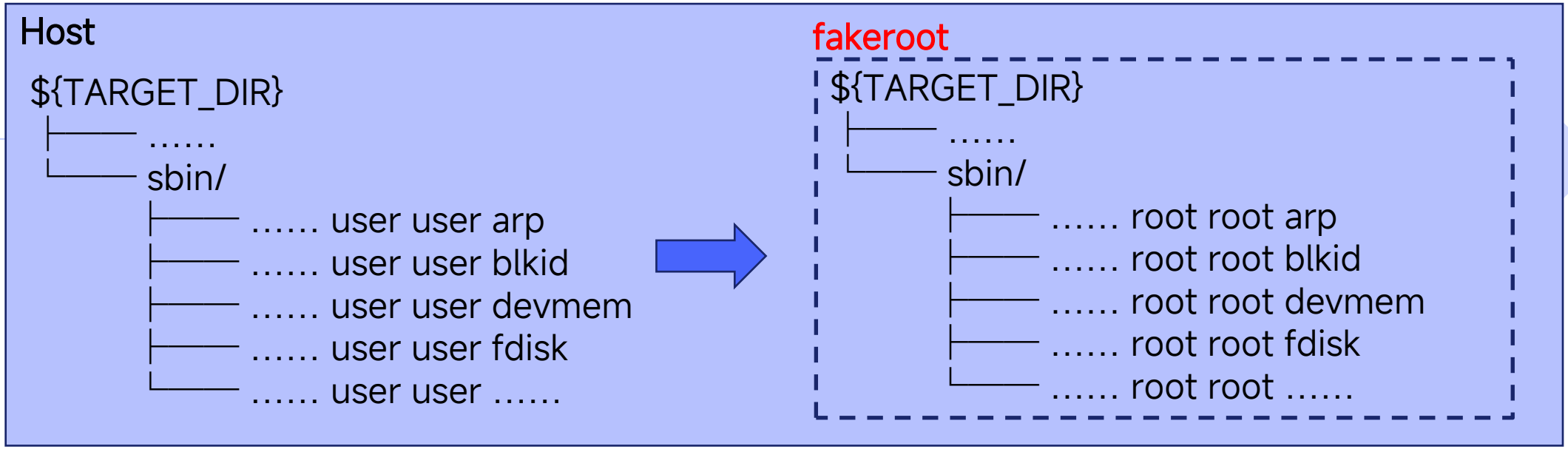
- fakeroot 是一个最初由 Debian 社区发起的，为了解决 Debian 打包过程中的安全问题而开发的工具软件。它能模拟出一个操作环境，使得普通用户在这个环境中运行程序时，这些程序会认为自己正以 root 身份运行，从而能执行一些通常需要 root 权限的操作（如修改文件的所有者、权限、创建设备节点等）。但这些修改仅在这个模拟环境中生效，不会影响真实的文件系统，一旦退出环境，所有模拟的效果都会消失。
- fakeroot 的主要用途有：软件打包、复杂脚本测试等。
- 构建 fakeroot 时依赖 acl，是因为现代 fakeroot 需要完整地模拟 root 环境，这包括了模拟文件系统的 ACL（访问控制列表）操作。

fakeroot 简介

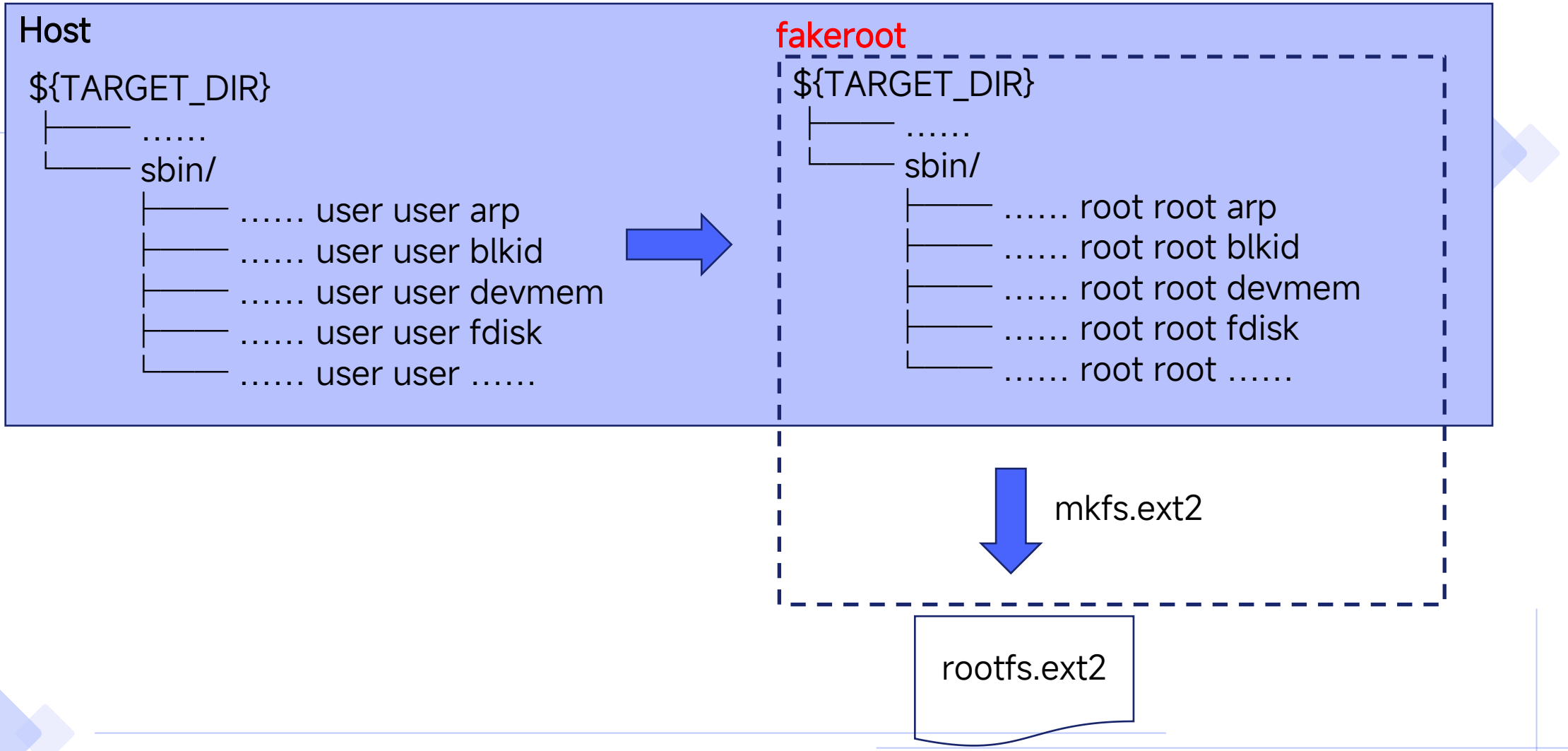
Host

```
${TARGET_DIR}
├──── ......
└──── sbin/
      ├──── ...... user user arp
      ├──── ...... user user blkid
      ├──── ...... user user devmem
      ├──── ...... user user fdisk
      └──── ...... user user ......
```

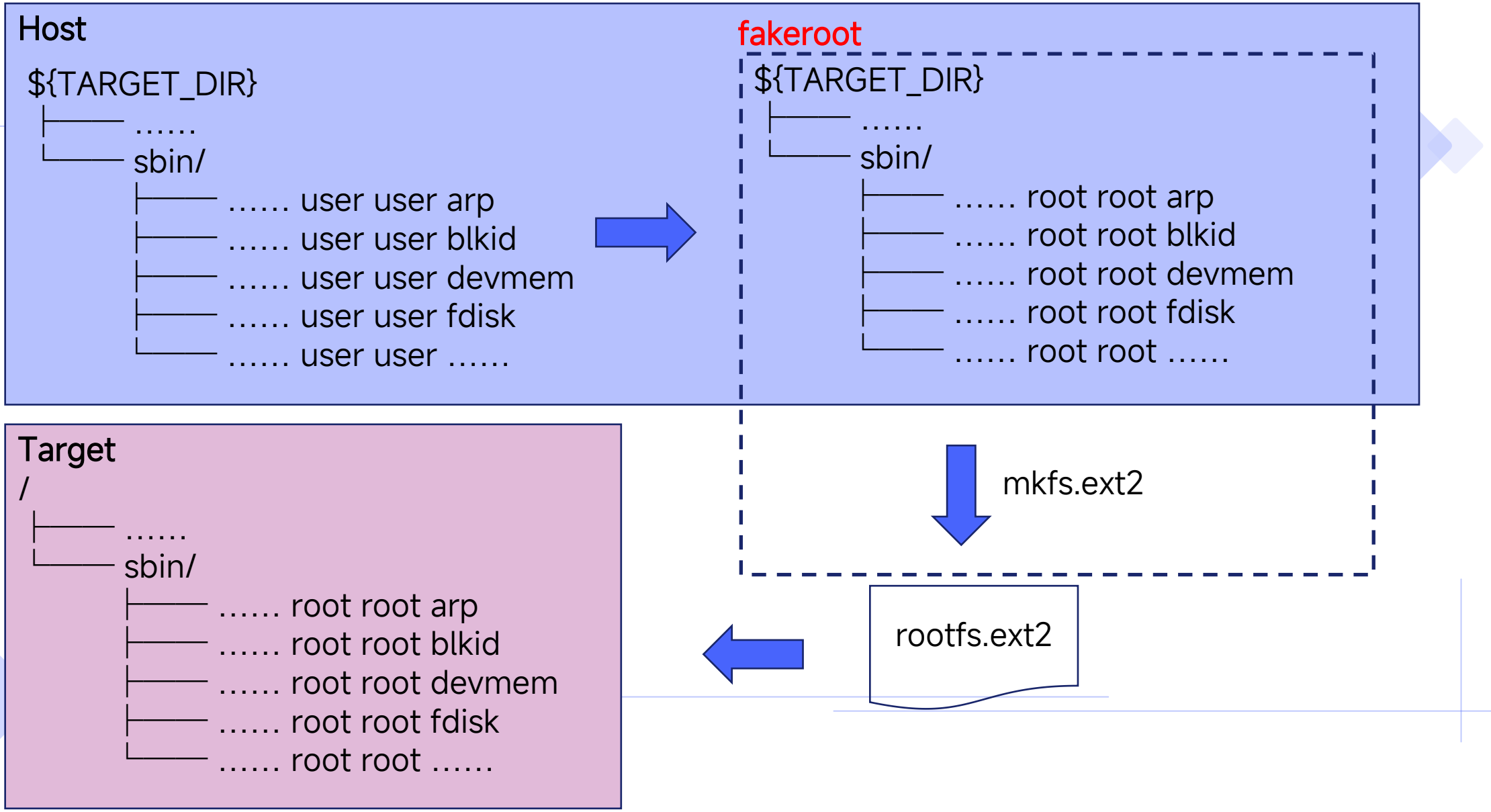
fakeroot 简介


Host
${TARGET_DIR}
├──── ......
└──── sbin/
├──── ..... user user arp
├──── ..... user user blkid
├──── ..... user user devmem
├──── ..... user user fdisk
└──── ..... user user ......
fakeroot
${TARGET_DIR}
├──── ......
└──── sbin/
├──── ..... root root arp
├──── ..... root root blkid
├──── ..... root root devmem
├──── ..... root root fdisk
└──── ..... root root ......

fakeroot 简介


Host
${TARGET_DIR}
├──── ......
└──── sbin/
├──── ..... user user arp
├──── ..... user user blkid
├──── ..... user user devmem
├──── ..... user user fdisk
└──── ..... user user ......
fakeroot
${TARGET_DIR}
├──── ......
└──── sbin/
├──── ..... root root arp
├──── ..... root root blkid
├──── ..... root root devmem
├──── ..... root root fdisk
└──── ..... root root ......
mkfs.ext2
rootfs.ext2

fakeroot 简介


Host
${TARGET_DIR}
├──── ……
└──── sbin/
├──── …… user user arp
├──── …… user user blkid
├──── …… user user devmem
├──── …… user user fdisk
└──── …… user user ……
fakeroot
${TARGET_DIR}
├──── ……
└──── sbin/
├──── …… root root arp
├──── …… root root blkid
├──── …… root root devmem
├──── …… root root fdisk
└──── …… root root ……
mkfs.ext2
rootfs.ext2
Target
/
├──── ……
└──── sbin/
├──── …… root root arp
├──── …… root root blkid
├──── …… root root devmem
├──── …… root root fdisk
└──── …… root root ……

fakeroot 的使用

基本语法

fakeroot [选项] [--] [命令]

例子：

```
$ fakeroot bash –c '
> chown -h -R 0:0 /home/rootfs
> mkfs.ext2 -d /home/rootfs rootfs.ext2
> '
```

fakeroot 的使用

基本语法

fakeroot [选项] [--] [命令]

例子：

```
$ fakeroot bash –c '
> chown -h -R 0:0 /home/rootfs
> tar -cf rootfs.tar /home/rootfs
> '
```

fakeroot 的使用

基本语法

fakeroot [选项] [--] [命令]

例子：

```
$ echo "#!/bin/sh" > /home/fakeroot
$ echo "chown -h -R 0:0 /home/rootfs " >> /home/fakeroot
$ echo "tar -cf rootfs.tar /home/rootfs " >> /home/fakeroot
$ chmod a+x /home/fakeroot
$ fakeroot /home/fakeroot
```

fakeroot - 制作步驟

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_header_sys_capability_h=no
ac_cv_func_capset=no CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --
disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

fakeroot - 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_header_sys_capability_h=no
ac_cv_func_capset=no CONFIG_SITE=/dev/n  /configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=  OST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gt  --disable-docs --
disable-documentation --disable-debug --\  --
disable-dependency-tracking"
```

去掉对 Linux capabilities 的依赖；减小构建结果体积和构建复杂度。

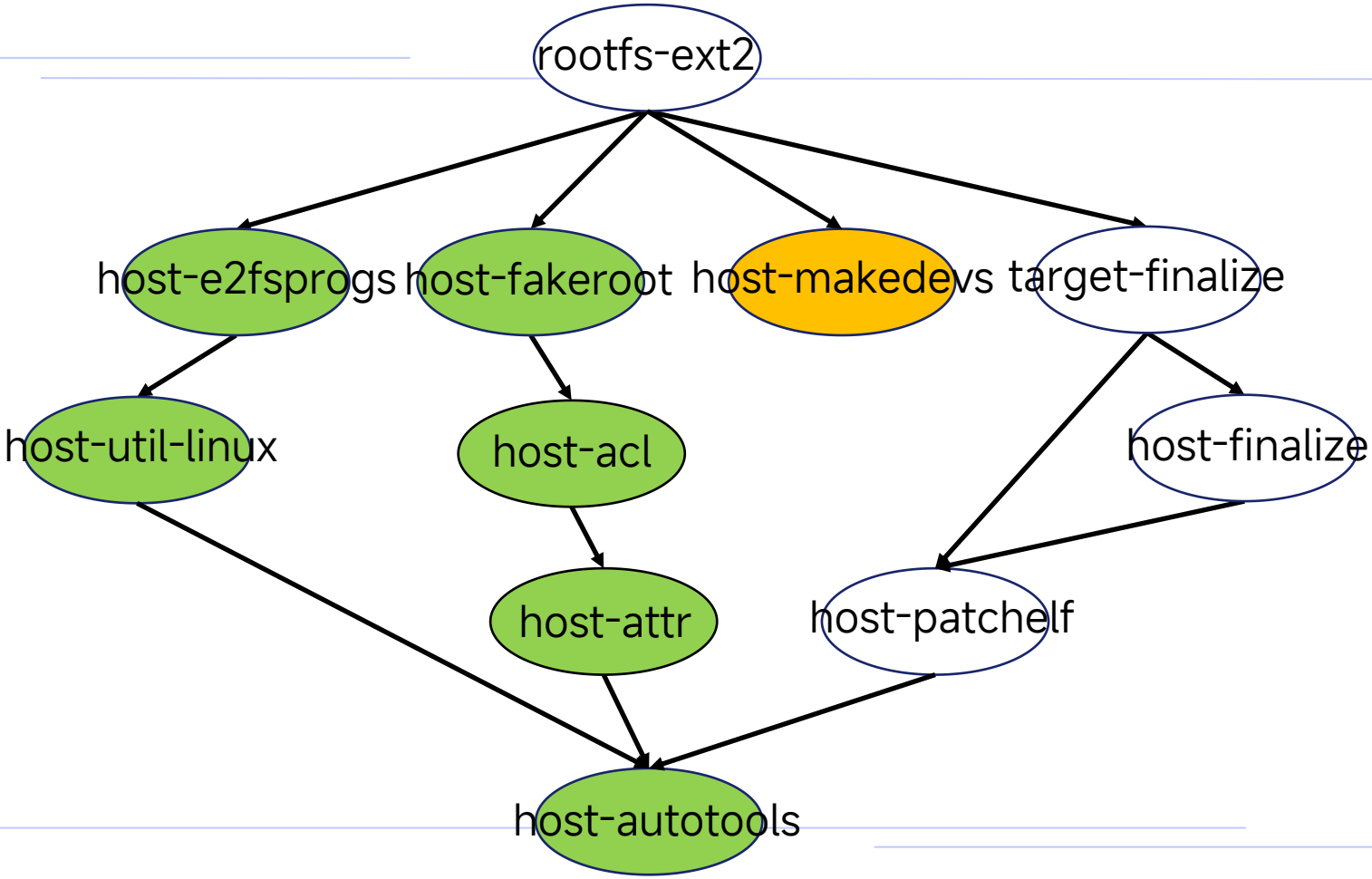
构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

makedevs - 制作步骤

- makedevs 是我们从 Buildroot 中借鉴过来的一个软件工具。在构建主机上被编译和运行。注意它并不是需要安装到目标系统内的软件包。
- 它的核心功能是根据一个设备表文件（device table），在目标根文件系统中批量创建节点并设置权限。主要用于在构建最终的 Linux 文件系统镜像时，快速、准确地创建 /dev 目录下的设备节点，以及设置关键文件和目录的所有权与权限。设备表文件的例子如下：

```
# 路径        类型  权限  用户  组      主设备号  次设备号  起始  增量  数量
/dev          d     755   root  root     -         -         -     -     -
/dev/console  c     600   root  root     5         1         -     -     -
/dev/ttyS0    c     600   root  dialout  4         64        0     1     4
```

- 具体使用可以参考本课程代码仓库的 package/rootfs-ext2/make.sh 中，例子如下：

```
$ makedevs -d full_devices_table.txt ${BUILD_DIR}/buildroot-fs/tar/target
```

makedevs - 制作步骤

无需配置

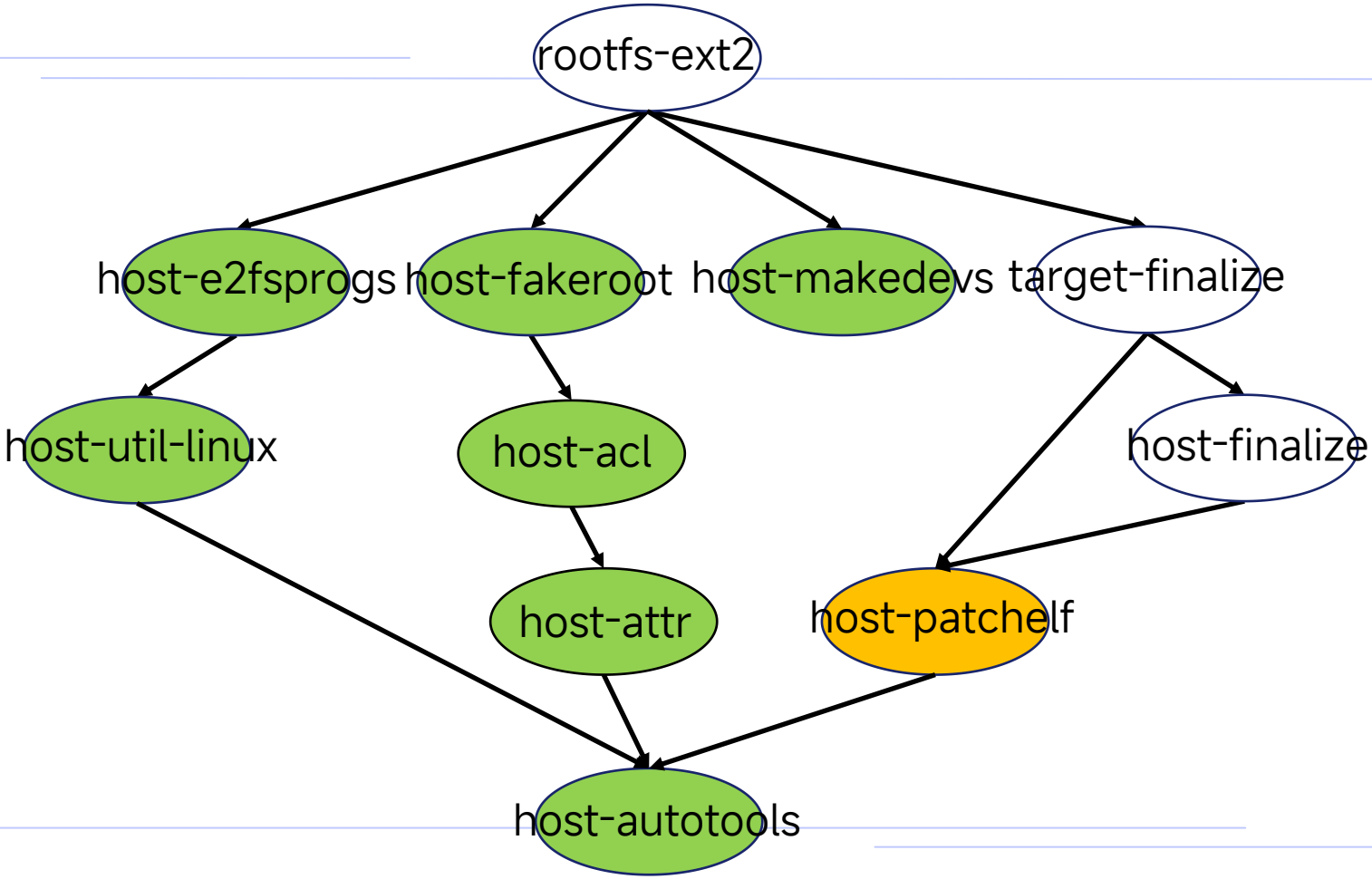
构建

```
/usr/bin/gcc -O2 -I${HOST_DIR}/include ${PKGBUILD_DIR}/makedevs.c -o
${PKGBUILD_DIR}/makedevs -L${HOST_DIR}/lib -Wl,-rpath,${HOST_DIR}/lib
```

安装 （install-host)

```
/usr/bin/install -D -m 755 ${PKGBUILD_DIR}/makedevs ${HOST_DIR}/bin/makedevs
```

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

patchelf 简介

https://github.com/NixOS/patchelf

- patchelf 是 NixOS 社区开发和维护的一个用于修改 ELF 可执行文件的实用工具。包括：
  - ✓ 修改可执行文件中记录的所使用的动态链接器（interpreter）的路径
  - ✓ 修改可执行程序或者共享库中搜索其依赖的共享库的路径 RPATH/RUNPATH
  - ✓ 调整共享库的 SONAME（共享对象名称）
- 我们在 fix-rpath 脚本中会调用 patchelf，而 fix-rpath 会在 host-finalize 以及 target-finalize 中被调用。

patchelf – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

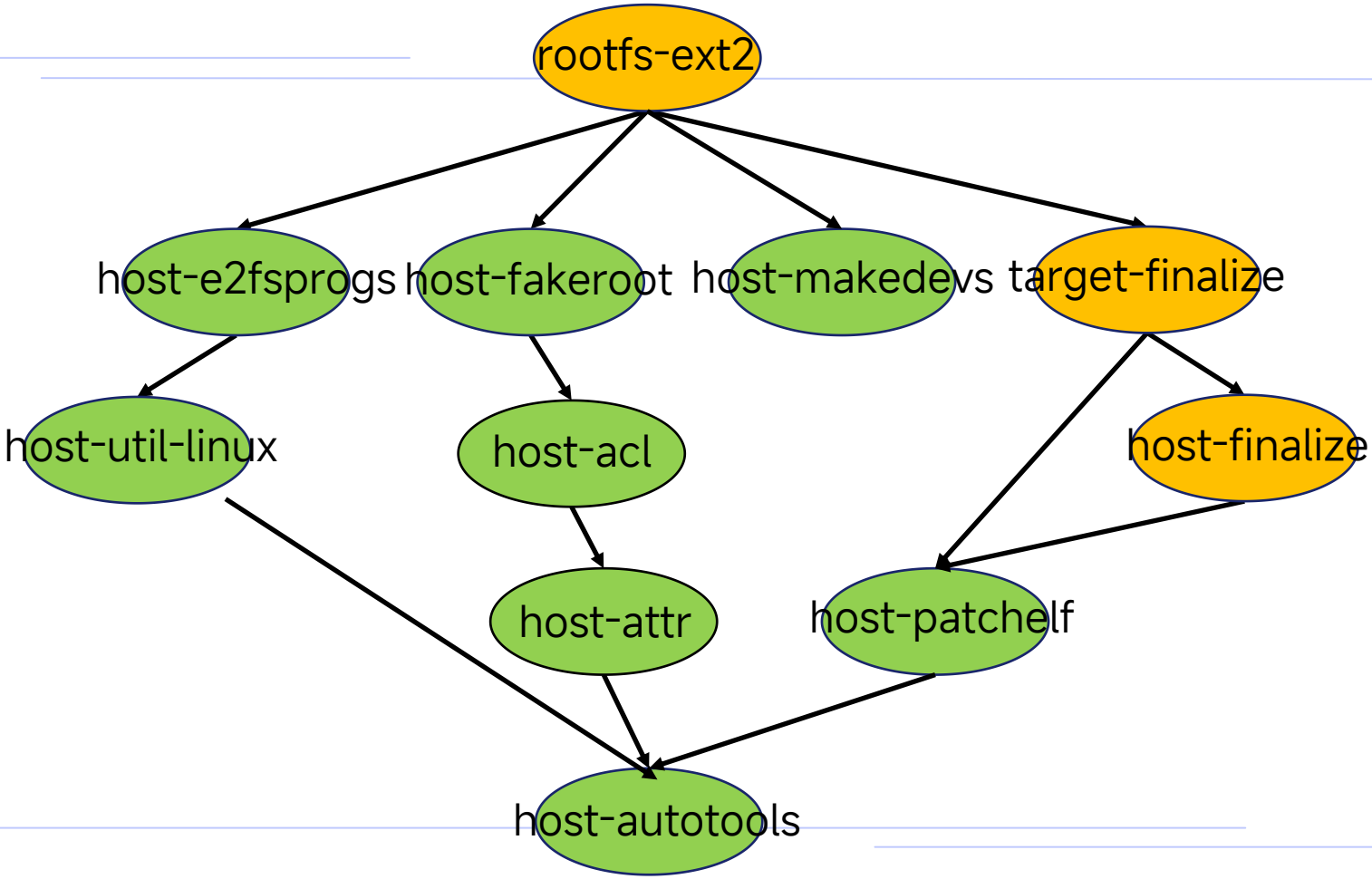
构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

制作根文件系统镜像


rootfs-ext2
host-e2fsprogs
host-fakeroot
host-makedevs
target-finalize
host-util-linux
host-acl
host-finalize
host-attr
host-patchelf
host-autotools

制作根文件系统镜像 – host-finalize

```
PATCHELF=${HOST_DIR}/bin/patchelf \
PARALLEL_JOBS=${MAXNUM_CPUS} \
PER_PACKAGE_DIR=${OUTPUT_DIR}/per-package \
${PROJECT_DIR}/support/scripts/fix-rpath host

PATCHELF=${HOST_DIR}/bin/patchelf \
PARALLEL_JOBS=${MAXNUM_CPUS} \
PER_PACKAGE_DIR=${OUTPUT_DIR}/per-package \
${PROJECT_DIR}/support/scripts/fix-rpath staging
```

制作根文件系统镜像 - target-finalize

```
/usr/bin/install -m 0755 -D ${PROJECT_DIR}/package/${PKGNAME}/ifupdown-scripts/nfs_check
${TARGET_DIR}/etc/network/nfs_check

......
```

更多处理不再赘述，具体参考仓库代码的package/target-finalize/make.sh

制作根文件系统镜像 – rootfs-ext2

```
……
mkdir -p ${BUILD_DIR}/buildroot-fs/ext2
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM ${TARGET_DIR}/ ${BUILD_DIR}/buildroot-fs/ext2/target
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
……
printf "        rm -f ${IMAGES_DIR}/rootfs.ext2\n       ${HOST_DIR}/sbin/mkfs.ext2 -d ${BUILD_DIR}/buildroot-fs/ext2/target -N 0 -m 5 -L
\"rootfs\" -I 256 -O ^64bit ${IMAGES_DIR}/rootfs.ext2 \"${ROOTFS_EXT2_SIZE}\" || { ret=\$?; echo \"*** Maybe you need to increase the
filesystem size (ROOTFS_EXT2_SIZE)\" 1>&2; exit \$ret; }\n" >> ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
chmod a+x ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOTDONTTRYCHOWN=1 ${HOST_DIR}/bin/fakeroot --
${BUILD_DIR}/buildroot-fs/ext2/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/ext2/target


……
mkdir -p ${BUILD_DIR}/buildroot-fs/tar
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM ${TARGET_DIR}/ ${BUILD_DIR}/buildroot-fs/tar/target
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/tar/fakeroot
……
printf "        (cd ${BUILD_DIR}/buildroot-fs/tar/target; find -print0 | LC_ALL=C sort -z | tar  --pax-
option=exthdr.name=%%d/PaxHeaders/%%f,atime:=0,ctime:=0 -cf ${IMAGES_DIR}/rootfs.tar --null --xattrs-include='*' --no-recursion -
T - --numeric-owner)\n" >> ${BUILD_DIR}/buildroot-fs/tar/fakeroot
chmod a+x ${BUILD_DIR}/buildroot-fs/tar/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOTDONTTRYCHOWN=1 ${HOST_DIR}/bin/fakeroot --
${BUILD_DIR}/buildroot-fs/tar/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/tar/target
……
```

制作根文件系统镜像 - rootfs-ext2

```
……
mkdir -p ${BUILD_DIR}/buildroot-fs/ext2
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM ${TARGET_DIR}/ ${BUILD_DIR}/buildroot-fs/ext2/target
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
……
printf "        rm -f ${IMAGES_DIR}/rootfs.ext2\n       ${HOST_DIR}/sbin/mkfs.ext2 -d ${BUILD_DIR}/buildroot-fs/ext2/target -N 0 -m 5 -L
\"rootfs\" -I 256 -O ^64bit ${IMAGES_DIR}/rootfs.ext2 \"${ROOTFS_EXT2_SIZE}\" || { ret=\$?; echo \"*** Maybe you need to increase the
filesystem size (ROOTFS_EXT2_SIZE)\" 1>&2; exit \$ret; }\n" >> ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
chmod a+x ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOTDONTTRYCHOWN=1 ${HOST_DIR}/bin/fakeroot --
${BUILD_DIR}/buildroot-fs/ext2/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/ext2/target


……
mkdir -p ${BUILD_DIR}/buildroot-fs/tar
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEI
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/tar/fakeroot
……
printf "        (cd ${BUILD_DIR}/buildroot-fs/tar/target; find -prin
option=exthdr.name=%%d/PaxHeaders/%%f,atime:=0,ctime:=0                                                                    -no-recursion -
T - --numeric-owner)\n" >> ${BUILD_DIR}/buildroot-fs/tar/fakero
chmod a+x ${BUILD_DIR}/buildroot-fs/tar/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOTDONTTRYCHOWN=1 ${HOST_DIR}/bin/fakeroot --
${BUILD_DIR}/buildroot-fs/tar/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/tar/target
……
```

编辑用于生成 rootfs.ext2 的 fakeroot 脚本，然后运行 fakeroot 命令执行该脚本生成 rootfs.ext2

制作根文件系统镜像 - rootfs-ext2

```
……
mkdir -p ${BUILD_DIR}/buildroot-fs/ext2
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM ${TARGET_DIR}/ ${BUILD_DIR}/buildroot-fs/ext2/target
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
……
printf "        rm -f ${IMAGES_DIR}/rootfs.ext2\n       ${HOST_DIR}/s                                                    -N 0 -m 5 -L
\"rootfs\" -I 256 -O ^64bit ${IMAGES_DIR}/rootfs.ext2 \"${ROOTFS                                                    increase the
filesystem size (ROOTFS_EXT2_SIZE)\" 1>&2; exit \$ret; }\n" >> ${B
chmod a+x ${BUILD_DIR}/buildroot-fs/ext2/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOT
${BUILD_DIR}/buildroot-fs/ext2/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/ext2/target

……
mkdir -p ${BUILD_DIR}/buildroot-fs/tar
rsync -auH --exclude=/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM ${TARGE                {BUILD_DIR}/buildroot-fs/tar/target
echo '#!/bin/sh' > ${BUILD_DIR}/buildroot-fs/tar/fakeroot
……
printf "        (cd ${BUILD_DIR}/buildroot-fs/tar/target; find -print0 | LC_AL    sort -z | tar  --pax-
option=exthdr.name=%%d/PaxHeaders/%%f,atime:=0,ctime:=0 -cf ${IMA    ES_DIR}/rootfs.tar --null --xattrs-include='*' --no-recursion -
T - --numeric-owner)\n" >> ${BUILD_DIR}/buildroot-fs/tar/fakeroot
chmod a+x ${BUILD_DIR}/buildroot-fs/tar/fakeroot
PATH="${HOST_DIR}/bin:${HOST_DIR}/sbin:${PATH}" FAKEROOTDONTTRYCHOWN=1 ${HOST_DIR}/bin/fakeroot --
${BUILD_DIR}/buildroot-fs/tar/fakeroot
rm -rf ${BUILD_DIR}/buildroot-fs/tar/target
……
```

编辑用于生成 rootfs.tar 的 fakeroot 脚本，然后运行 fakeroot 命令执行该脚本生成 rootfs.tar

运行

```
$ cd ${IMAGES_DIR}
$ ls -l
total 99128
-rw-r--r-- 1 wangchen wangchen   273464 Dec 29 18:07 fw_jump.bin
-rw-r--r-- 1 wangchen wangchen 22122496 Dec 29 18:09 Image
-rw-rw-r-- 1 wangchen wangchen 26591232 Dec 29 18:10 initrd.img
-rw-rw-r-- 1 wangchen wangchen 62914560 Dec 29 18:10 rootfs.ext2
-rw-rw-r-- 1 wangchen wangchen 27852800 Dec 29 18:10 rootfs.tar
-rwxrwxr-x 1 wangchen wangchen      207 Dec 29 18:10 start-qemu-initramfs.sh
-rwxrwxr-x 1 wangchen wangchen      265 Dec 29 18:10 start-qemu.sh
$ ./start-qemu.sh

OpenSBI v1.6

   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
......
[    0.000000] Linux version 6.12.47 ……
[    0.000000] Machine model: riscv-virtio,qemu
......
[    0.504811] Run /sbin/init as init process
[    0.666292] EXT4-fs (vda): re-mounted 52852f9d-5797-429e-98fc-cbd063a24159.
Saving 256 bits of non-creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: udhcpc: started, v1.37.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.0.2.15, server 10.0.2.2
udhcpc: lease of 10.0.2.15 obtained from 10.0.2.2, lease time 86400
deleting routers
adding dns 10.0.2.3
OK
Starting crond: OK

Welcome to Build Linux System From Scratch
buildlinux login:
```

运行

```
$ cd ${IMAGES_DIR}
$ ls -l
total 99128
-rw-r--r-- 1 wangchen wangchen   273464 Dec 29 18:07 fw_jump.bin
-rw-r--r-- 1 wangchen wangchen 22122496 Dec 29 18:09 Image
-rw-rw-r-- 1 wangchen wangchen 26591232 Dec 29 18:10 initrd.img
-rw-rw-r-- 1 wangchen wangchen 62914560 Dec 29 18:10 rootfs.ext2
-rw-rw-r-- 1 wangchen wangchen 27852800 Dec 29 18:10 rootfs.tar
-rwxrwxr-x 1 wangchen wangchen      207 Dec 29 18:10 start-qemu-initramfs.sh
-rwxrwxr-x 1 wangchen wangchen      265 Dec 29 18:10 start-qemu.sh
$ ./start-qemu.sh

OpenSBI v1.6

   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
......
[    0.000000] Linux version 6.12.47 ……
[    0.000000] Machine model: riscv-virtio,qemu
......
[    0.504811] Run /sbin/init as init process
[    0.666292] EXT4-fs (vda): re-mounted 52852f9d-5797-429e-98fc-cbd063a24159.
Saving 256 bits of non-creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: udhcpc: started, v1.37.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.0.2.15, server 10.0.2.2
udhcpc: lease of 10.0.2.15 obtained from 10.0.2.2, lease time 86400
deleting routers
adding dns 10.0.2.3
OK
Starting crond: OK

Welcome to Build Linux System From Scratch
buildlinux login:
```

qemu-system-riscv64 -M virt -m 256M -nographic -bios fw_jump.bin -kernel Image -drive file=rootfs.ext2,format=raw,id=hd0 -device virtio-blk-device,drive=hd0 -append \"nokaslr root=/dev/vda rw console=ttyS0\" -netdev user,id=net0 -device virtio-net-device,netdev=net0

运行

```
$ cd ${IMAGES_DIR}
$ ls -l
total 99128
-rw-r--r-- 1 wangchen wangchen   273464 Dec 29 18:07 fw_jump.bin
-rw-r--r-- 1 wangchen wangchen 22122496 Dec 29 18:09 Image
-rw-rw-r-- 1 wangchen wangchen 26591232 Dec 29 18:10 initrd.img
-rw-rw-r-- 1 wangchen wangchen 62914560 Dec 29 18:10 rootfs.ext2
-rw-rw-r-- 1 wangchen wangchen 27852800 Dec 29 18:10 rootfs.tar
-rwxrwxr-x 1 wangchen wangchen      207 Dec 29 18:10 start-qemu-initramfs.sh
-rwxrwxr-x 1 wangchen wangchen      265 Dec 29 18:10 start-qemu.sh
$ ./start-qemu.sh

OpenSBI v1.6

   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
......
[    0.000000] Linux version 6.12.47 ……
[    0.000000] Machine model: riscv-virtio,qemu
......
[    0.504811] Run /sbin/init as init process
[    0.666292] EXT4-fs (vda): re-mounted 52852f9d-5797-429e-98fc-cbd063a24159.
Saving 256 bits of non-creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: udhcpc: started, v1.37.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.0.2.15, server 10.0.2.2
udhcpc: lease of 10.0.2.15 obtained from 10.0.2.2, lease time 86400
deleting routers
adding dns 10.0.2.3
OK
Starting crond: OK

Welcome to Build Linux System From Scratch
buildlinux login:
```

qemu-system-riscv64 -M virt -m 256M -nographic -bios fw_jump.bin -kernel Image -drive file=rootfs.ext2,format=raw,id=hd0 -device virtio-blk-device,drive=hd0 -append \"nokaslr root=/dev/vda rw console=ttyS0\" -netdev user,id=net0 -device virtio-net-device,netdev=net0

运行

```
$ cd ${IMAGES_DIR}
$ ls -l
total 99128
-rw-r--r-- 1 wangchen wangchen   273464 Dec 29 18:07 fw_jump.bin
-rw-r--r-- 1 wangchen wangchen 22122496 Dec 29 18:09 Image
-rw-rw-r-- 1 wangchen wangchen 26591232 Dec 29 18:10 initrd.img
-rw-rw-r-- 1 wangchen wangchen 62914560 Dec 29 18:10 rootfs.ext2
-rw-rw-r-- 1 wangchen wangchen 27852800 Dec 29 18:10 rootfs.tar
-rwxrwxr-x 1 wangchen wangchen      207 Dec 29 18:10 start-qemu-initramfs.sh
-rwxrwxr-x 1 wangchen wangchen      265 Dec 29 18:10 start-qemu.sh
$ ./start-qemu.sh

OpenSBI v1.6

   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
......
[    0.000000] Linux version 6.12.47 ……
[    0.000000] Machine model: riscv-virtio,qemu
......
[    0.504811] Run /sbin/init as init process
[    0.666292] EXT4-fs (vda): re-mounted 52852f9d-5797-429e-98fc-cbd063a24159.
Saving 256 bits of non-creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: udhcpc: started, v1.37.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.0.2.15, server 10.0.2.2
udhcpc: lease of 10.0.2.15 obtained from 10.0.2.2, lease time 86400
deleting routers
adding dns 10.0.2.3
OK
Starting crond: OK

Welcome to Build Linux System From Scratch
buildlinux login:
```

02 基于 SysV 改进 Init 系统

Init System 简介

Init System（初始化系统）即 Linux 操作系统中 **第一个启动的进程**（PID=1）。其主要职责包括：

- **启动系统服务**：在系统启动时，按照依赖关系启动各种系统服务（如网络、日志、调度任务等）。
- **管理服务状态**：在系统运行期间，允许用户查看、启动、停止、重启服务。
- **守护进程**：确保关键服务在意外退出时能自动重启。
- **设置运行级别**：定义不同的系统状态（如单用户模式、多用户图形界面模式等）。
- **回收孤儿进程**：清理那些父进程已退出的“孤儿进程”。

主流 Init System

类型	介绍	优点	缺点
SysVinit（System V Init）	最经典、历史最悠久的 init 系统，源自 UNIX System V；用于一些早期的主要发行版：早期的 Red Hat、CentOS、Debian、Ubuntu（在 Upstart 出现之前）	简单、直观、稳定。 基于纯文本的 Bash 脚本，易于理解和手动修改	串行启动，效率低 对依赖的管理能力弱，完全需要手工调整 对现代硬件支持不佳，譬如对热插拔设备处理能力弱
Upstart	由 Canonical 公司为 Ubuntu 开发，旨在解决 SysVinit 的瓶颈；用于6.10 到 14.10 期间的 Ubuntu，Red Hat Enterprise Linux 6；早期的 Chrome OS	支持并行启动 更好的依赖和事件（热插拔）处理	配置相对复杂
Systemd	更统一、更高效的系统和服务管理平台; 目前绝大多数主流 Linux 发行版的默认 init 系统	支持并行启动 统一的配置文件和管理命令 更丰富的管理功能	相对庞大和复杂。

Sysvinit 简介

https://codeberg.org/thejessesmith/sysvinit

SysV init 是一个经典的初始化程序，适用于 GNU/Linux 和其他 UNIX/POSIX 系统。它的设计目标是体积小、操作简单，并且不干扰系统运行。其特点包括：

- **运行级别**：定义了系统处于何种状态。每个级别对应一组需要启动或停止的服务。
- **串行启动**：服务严格按照启动脚本 /etc/inittab 定义的顺序，一个接一个地启动。优点是逻辑清晰，缺点是启动速度慢，无法利用多核CPU的并发优势。
- **基于 Shell 脚本**：所有服务的启动、停止、重启逻辑都写在 /etc/init.d/ 目录下的 Shell 脚本中。
- **依赖关系简单**：通过脚本名字（S/K 后的数字顺序）来隐式管理依赖。

Sysvinit 简介

https://codeberg.org/thejessesmith/sysvinit

SysV init 是一个经典的初始化程序，适用于 GNU/Linux 和其他 UNIX/POSIX 系统。它的设计目标是体积小、操作简单，并且不干扰系统运行。其特点包括：

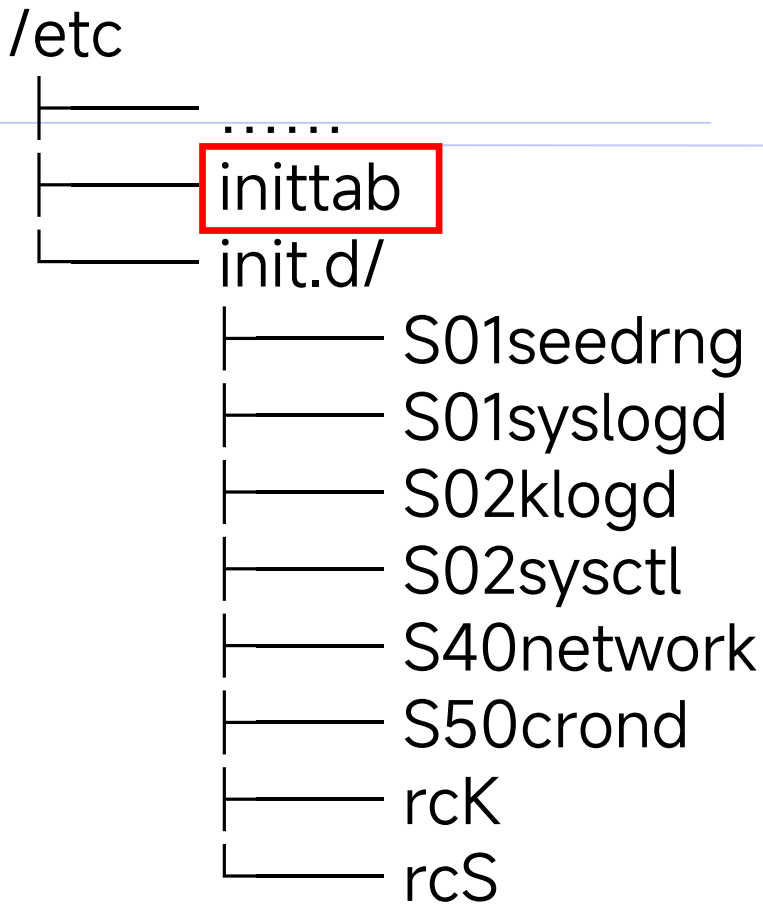
- **运行级别**：定义了系统处
  止的服务。
- **串行启动**：服务
  地启动。优点是逻辑
  发优势。
- **基于 Shell 脚本**：所有
  目录下的 Shell 脚本中。
- **依赖关系简单**：通过脚

级别	含义	例子
0	停机	init 0 # poweroff
1	单用户模式（救援模式，无网络，仅 root）	
2	多用户模式，无网络文件系统（NFS）	
3	完整的多用户文本模式（服务器标准运行级别）	
4	用户自定义（通常未使用）	
5	多用户图形界面模式（桌面标准运行级别）	
6	重启	init 6 # reboot

Sysvinit 简介

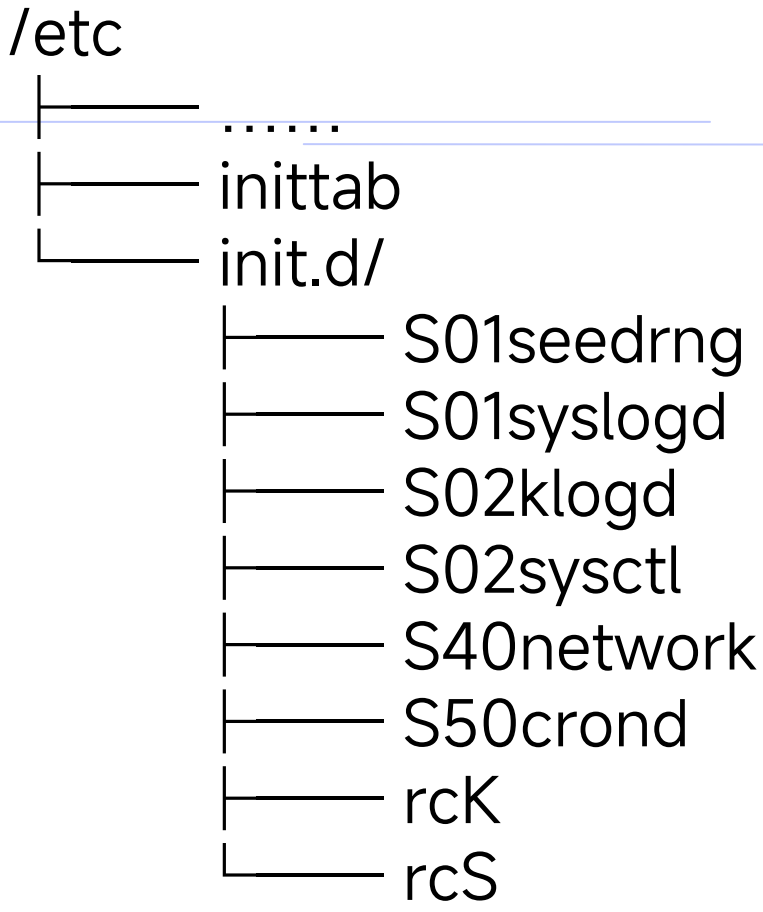
```
/etc
├──── ......
├──── inittab
└──── init.d/
        ├──── S01seedrng
        ├──── S01syslogd
        ├──── S02klogd
        ├──── S02sysctl
        ├──── S40network
        ├──── S50crond
        ├──── rcK
        └──── rcS
```

Sysvinit 简介




```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
......
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

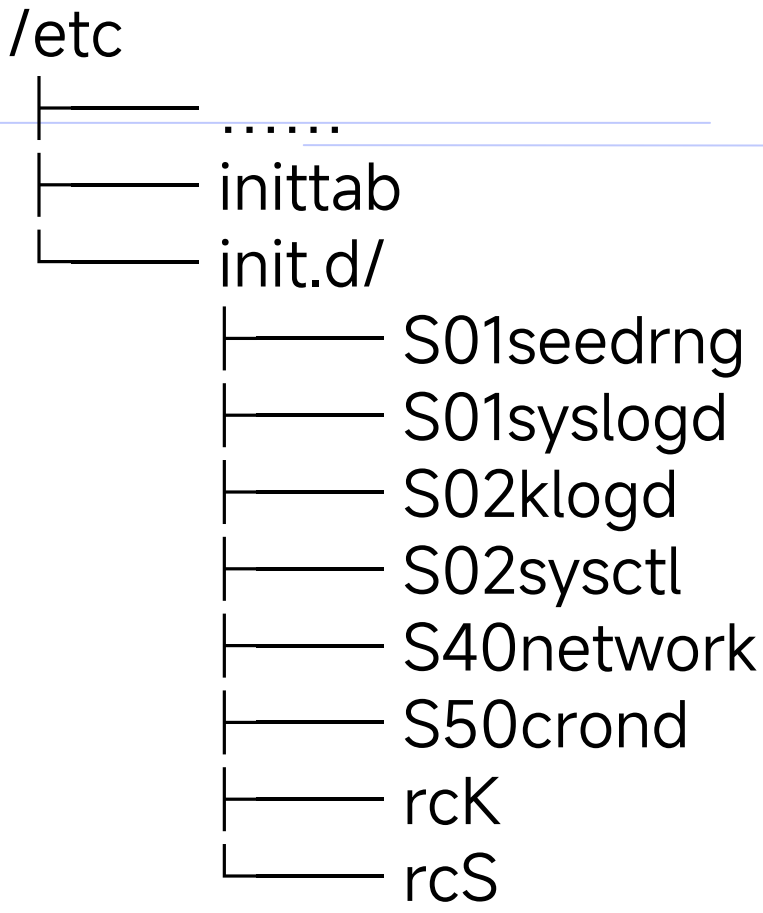
Sysvinit 简介



id : runlevels : action : process

```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
……
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

Sysvinit 简介

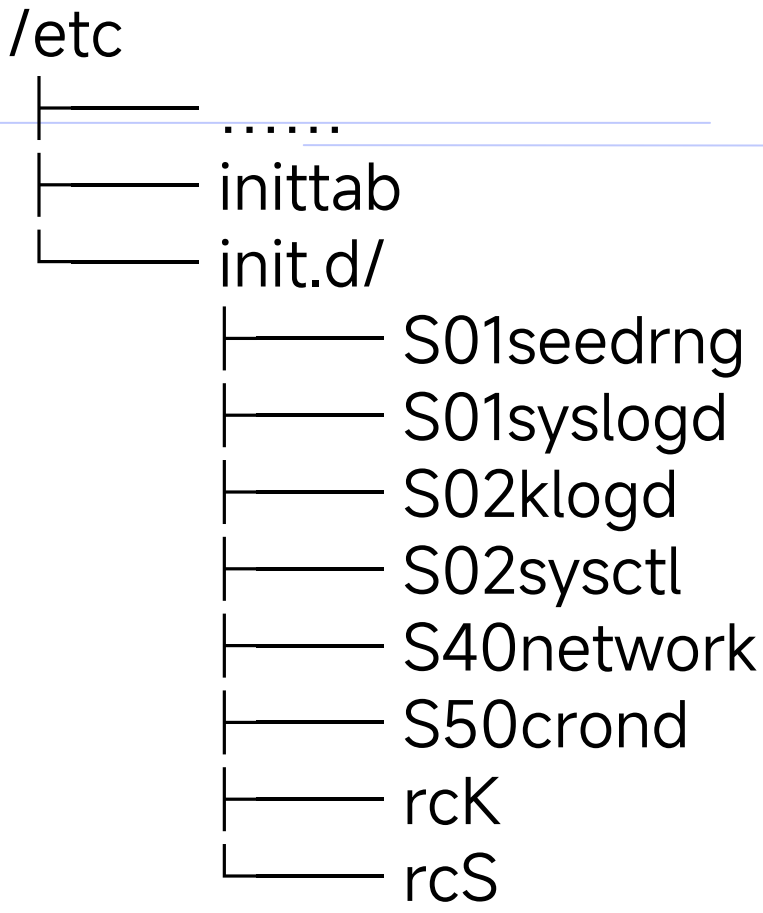


id:3:initdefault:

id : runlevels : action : process

```
si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
......
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

Sysvinit 简介

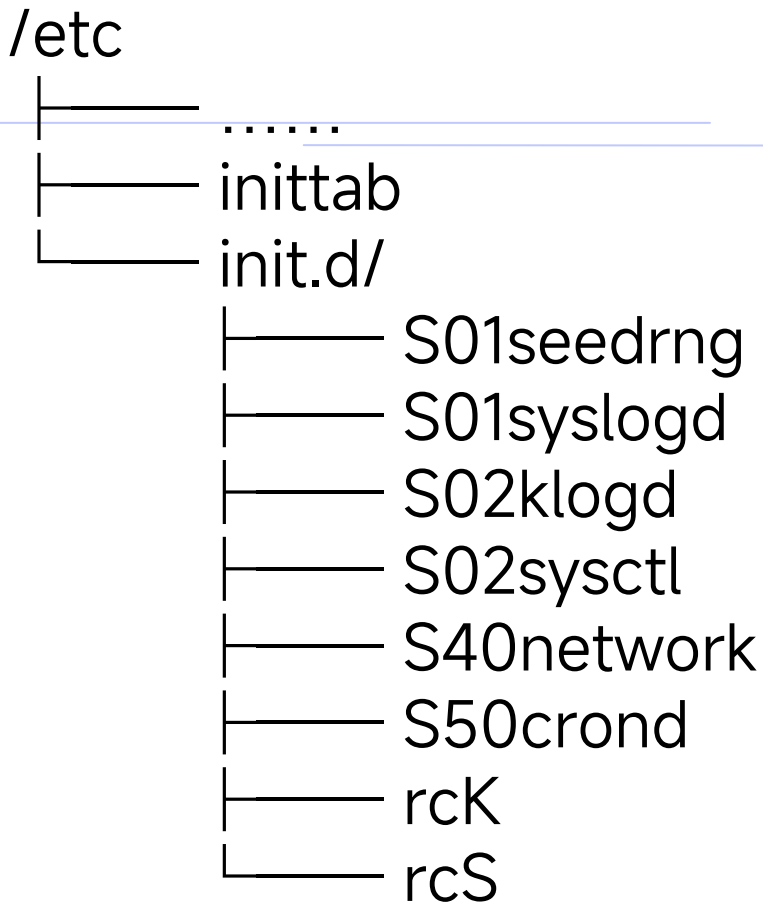


id:3:initdefault:

id : runlevels : action : process

```
si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
......
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

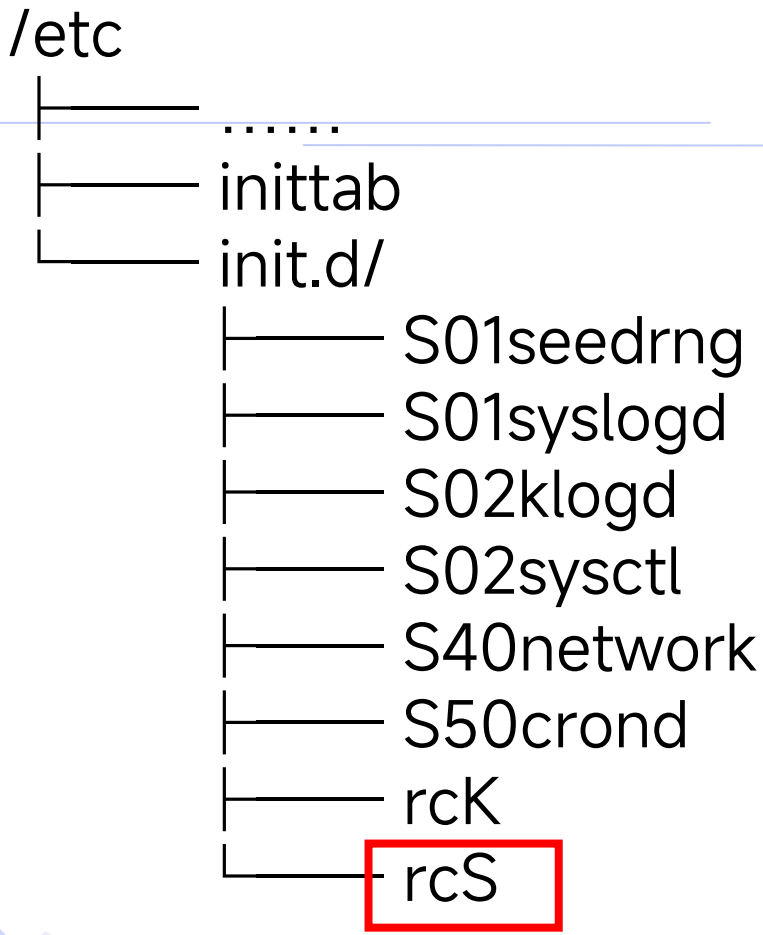
Sysvinit 简介



id : runlevels : action : process

```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
......
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

Sysvinit 简介




```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
……
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

                                                    SERIAL
```

```
#!/bin/sh
# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do
    ……
    case "$i" in
        ……
        *)
            # No sh extension, so fork subprocess.
            $i start
            ;;
    esac
done
```

Sysvinit 简介

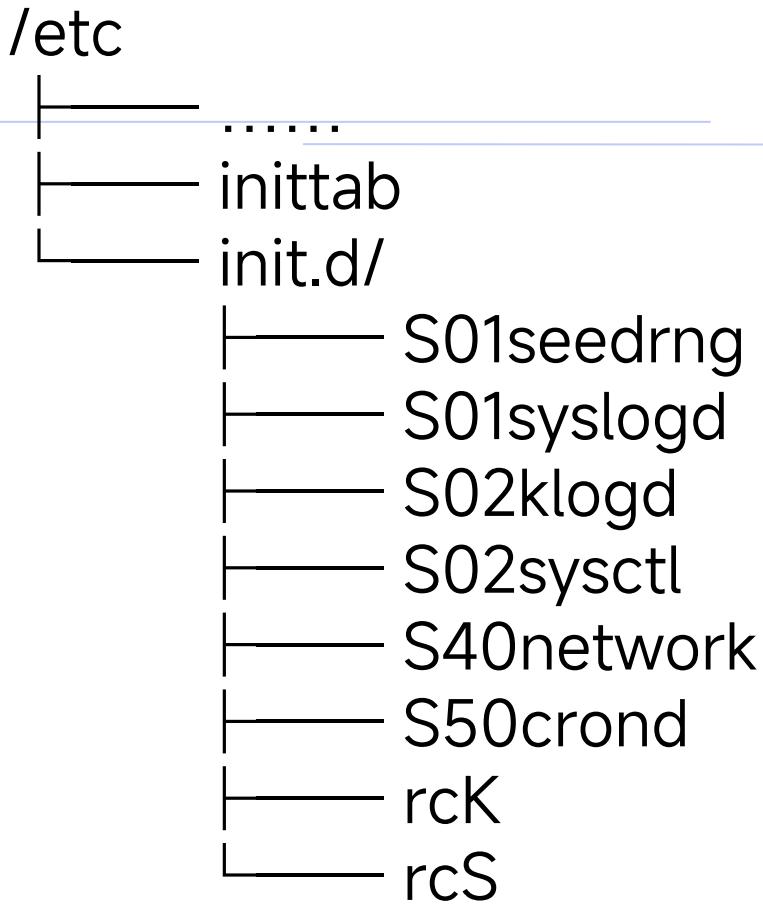
```
/etc
├── ……
├── inittab
└── init.d/
    ├── S01seedrng
    ├── S01syslogd
    ├── S02klogd
    ├── S02sysctl
    ├── S40network
    ├── S50crond
    ├── rcK
    └── rcS
```

```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
……
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
```

```
[    0.514047] Run /sbin/init as init process
INIT: version 3.14 booting
INIT: No inittab.d directory found
[    0.746229] EXT4-fs (vda): re-mounted 049ea23c-fd55-4f69-9ac5-3319b0d3ef01.
INIT: Entering runlevel: 3
Seeding 256 bits without crediting
Saving 256 bits of non-creditable seed for next boot
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Starting network: udhcpc: started, v1.37.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.0.2.15, server 10.0.2.2
udhcpc: lease of 10.0.2.15 obtained from 10.0.2.2, lease time 86400
deleting routers
adding dns 10.0.2.3
OK
Starting crond: OK
```

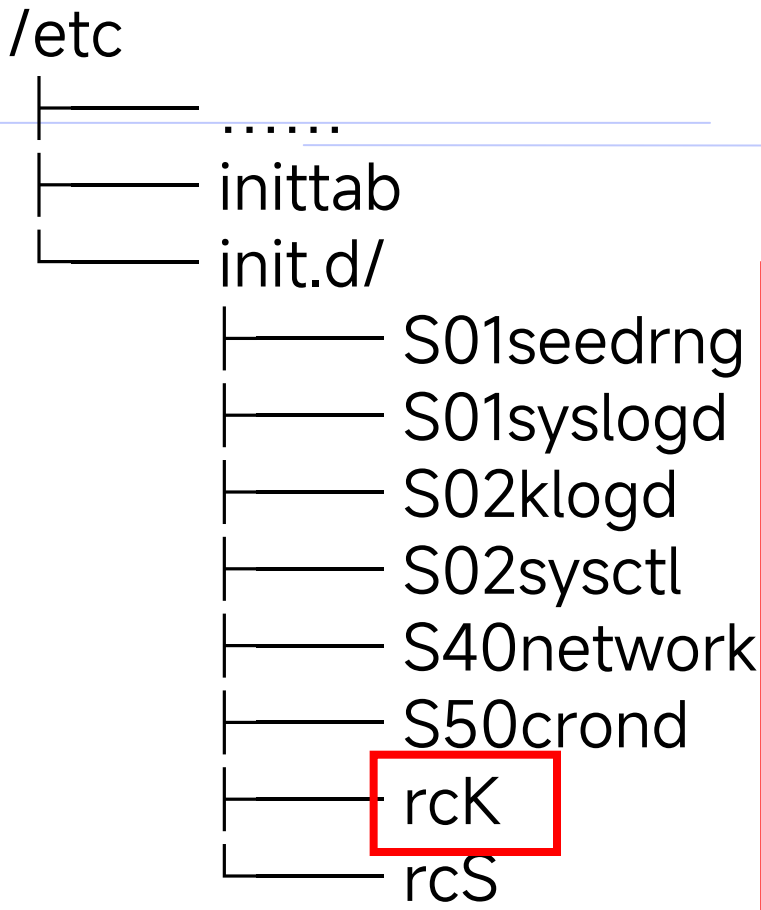
```
    esac
done
```

Sysvinit 简介




```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
……
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

Sysvinit 简介




```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
……
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS

                                                    _SERIAL
```

```
#!/bin/sh
# Stop all init scripts in /etc/init.d
# executing them in reversed numerical order.
#
for i in $(ls -r /etc/init.d/S??*) ;do
    ……
    *)
        # No sh extension, so fork subprocess.
        $i stop
        ;;
    esac
done
```

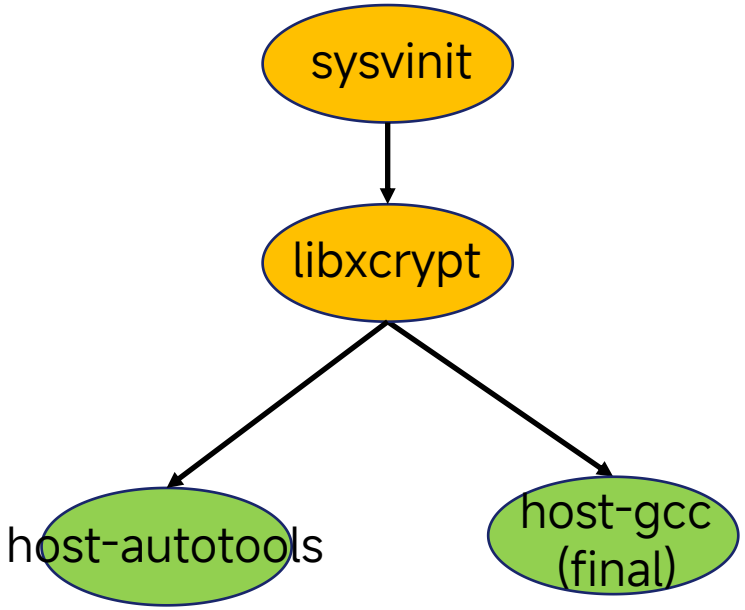
Sysvinit 简介

```
/etc
├── ......
├── inittab
└── init.d/
    ├── S01seedrng
    ├── S01syslogd
    ├── S02klogd
    ├── S02sysctl
    ├── S40network
    ├── S50crond
    ├── rcK
    └── rcS
```

```
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts /dev/shm
......
si9::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr 2>/dev/null
si10::sysinit:/bin/hostname -F /etc/hostname
rcS:12345:wait:/etc/init.d/rcS
```

```
# poweroff

Broadcast message from root@buildlinux (console) (Tue Dec 16 01:40:05 2025):

The system is going down for system halt NOW!
INIT: Switching to runlevel: 0
INIT: No inittab.d directory found
Stopping crond: stopped /usr/sbin/crond (pid 138)
OK
Stopping network: OK
Stopping klogd: OK
Stopping syslogd: stopped /sbin/syslogd (pid 89)
OK
Seeding 256 bits without crediting
Saving 256 bits of non-creditable seed for next boot
umount: tmpfs busy - remounted read-only
umount: devtmpfs busy - remounted read-only
[   35.829584] EXT4-fs (vda): re-mounted 049ea23c-fd55-4f69-9ac5-3319b0d3ef01 ro.
[   35.860792] reboot: Power down
```

Sysvinit 制作过程



libxcrypt 简介

https://github.com/besser82/libxcrypt

- libxcrypt 是一个开源密码哈希库，是原 Glibc 中的 libcrypt 库的一个兼容替代版本。
- libxcrypt 项目独立于 Glibc。
- libxcrypt 支持更多现代哈希算法（yescrypt、scrypt、bcrypt、SHA-512、SHA-256 等）；也支持一些旧的加密算法（DES、MD5 等）
- 为 login、passwd 等系统程序提供安全的密码哈希功能。
- sysvinit 中会调用 crypt() 函数，导致构建和运行都依赖于 libxcrypt 库。

libxcrypt - 制作步驟

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no   CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-
gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-
prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --
enable-ipv6 --disable-nls --disable-static --enable-shared  --disable-werror --disable-
obsolete_api"
```

libxcrypt - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no   CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-
gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-
prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --
enable-ipv6 --disable-nls --disable-static --enable-shared  --disable-werror --disable-
obsolete_api"
```

libxcrypt - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"
```

sysvinit - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no   CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-
gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-
prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --
enable-ipv6 --disable-nls --disable-static --enable-shared  --disable-werror --disable-
obsolete_api"
```

sysvinit - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no   CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-
gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-
prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --
enable-ipv6 --disable-nls --disable-static --enable-shared  --disable-werror --disable-
obsolete_api"
```

sysvinit - 制作步骤

直接构建，无需配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no /usr/bin/make -j${MAXNUM_CPUS}
SYSROOT=${STAGING_DIR} VERSION=3.14 -C ${PKGBUILD_DIR}/src"
```

安装 （install-target)

```
for x in halt init shutdown killall5; do
        /usr/bin/install -D -m 0755 ${PKGBUILD_DIR}/src/$x ${TARGET_DIR}/sbin/$x || exit 1;
done

/usr/bin/install -D -m 0644 ${PROJECT_DIR}/package/sysvinit/inittab ${TARGET_DIR}/etc/inittab

ln -sf /sbin/halt ${TARGET_DIR}/sbin/reboot

ln -sf /sbin/halt ${TARGET_DIR}/sbin/poweroff

ln -sf killall5 ${TARGET_DIR}/sbin/pidof
```

03

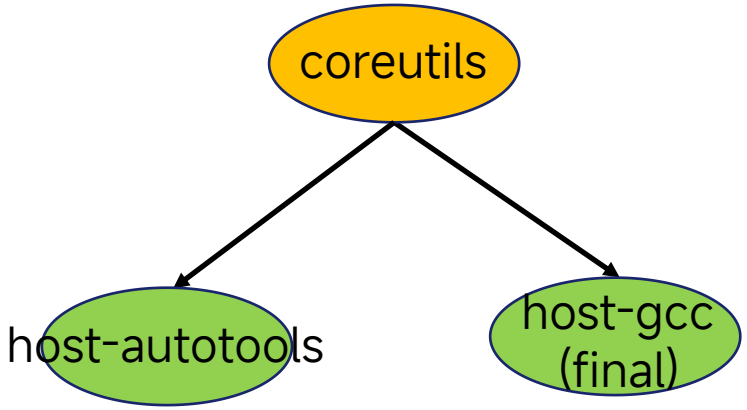
安装更完善的系统管理工具(Coreutils)

coreutils 简介

https://www.gnu.org/software/coreutils

- coreutils（GNU Core Utilities）是 GNU 操作系统中最基础、最核心的软件包之一，它提供了 Unix/Linux 系统中必不可少的命令行工具集。
- coreutils 的特色：遵循 POSIX 标准；跨平台支持（Linux、macOS、Windows（通过 Cygwin/WSL））以及多语言支持。
- coreutils 包含三类主要工具：
  - 文件操作类：进一步细分为：
    - 基本操作：ls、cp、mv、rm、mkdir、rmdir、touch；
    - 文件查看：cat、tac、head、tail、more、less；
    - 文件属性：chmod、chown、chgrp、stat
  - 文本处理类：sort、uniq、wc、cut、paste、tr
  - 系统信息类：date、echo、pwd、whoami、hostname
- coreutils 是我们运行 Shell脚本的基础，也是系统管理的必备工具箱，完美体现了 Unix 的设计哲学：“一个工具做好一件事”。

Coreutils 制作过程



coreutils - 制作步驟

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes ac_cv_have_decl_malloc=yes
gl_cv_func_malloc_0_nonnull=yes ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  ac_cv_c_restrict=no ac_cv_func_chown_works=yes ac_cv_func_fstatat=yes
ac_cv_func_getdelim=yes ac_cv_func_getgroups=yes ac_cv_func_getgroups_works=yes
ac_cv_func_getloadavg=no ac_cv_func_strnlen_working=yes ac_cv_have_decl_strerror_r=yes
ac_cv_have_decl_strnlen=yes ac_cv_lib_getloadavg_getloadavg=no ac_cv_lib_util_getloadavg=no
ac_fsusage_space=yes ac_use_included_regex=no am_cv_func_working_getline=yes
fu_cv_sys_stat_statfs2_bsize=yes gl_cv_func_getcwd_null=yes gl_cv_func_getcwd_path_max=yes
gl_cv_func_link_follows_symlink=no gl_cv_func_lstat_dereferences_slashed_symlink=yes
gl_cv_func_re_compile_pattern_working=yes gl_cv_func_svid_putenv=yes
gl_cv_func_working_mkstemp=yes gl_cv_func_working_utimes=yes
gl_cv_macro_MB_CUR_MAX_good=yes gl_cv_have_proc_uptime=yes utils_cv_localtime_cache=no
PERL=missing MAKEINFO=true INSTALL_PROGRAM=/usr/bin/install CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --
prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-
doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-
fop=no --disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-rpath  --disable-single-binary --disable-acl --disable-xattr --disable-libcap --without-selinux"
```

coreutils - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes ac_cv_have_decl_malloc=yes
gl_cv_func_malloc_0_nonnull=yes ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  ac_cv_c_restrict=no ac_cv_func_chown_works=yes ac_cv_func_fstatat=yes
ac_cv_func_getdelim=yes ac_cv_func_getgroups=yes ac_cv_func_getgroups_works=yes
ac_cv_func_getloadavg=no ac_cv_func_strnlen_working=yes ac_cv_have_decl_strerror_r=yes
ac_cv_have_decl_strnlen=yes ac_cv_lib_getloadavg_getloadavg=no ac_cv_lib_util_getloadavg=no
ac_fsusage_space=yes ac_use_included_regex=no am_cv_func_working_getline=yes
fu_cv_sys_stat_statfs2_bsize=yes gl_cv_func_getcwd_null=yes gl_cv_func_getcwd_path_max=yes
gl_cv_func_link_follows_symlink=no gl_cv_func_lstat_dereferences_slashed_symlink=yes
gl_cv_func_re_compile_pattern_working=yes gl_cv_func_svid_putenv=yes
gl_cv_func_working_mkstemp=yes gl_cv_func_working_utimes=yes
gl_cv_macro_MB_CUR_MAX_good=yes gl_cv_have_proc_uptime=yes utils_cv_localtime_cache=no
PERL=missing MAKEINFO=true INSTALL_PROGRAM=/usr/bin/install CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --
prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-
doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-
fop=no --disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-rpath  --disable-single-binary --disable-acl --disable-xattr --disable-libcap --without-selinux"
```

coreutils - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

COREUTILS_BIN_PROGS="base64 cat chgrp chmod chown cp date dd df dir echo false \
        kill link ln ls mkdir mknod mktemp mv nice printenv pwd rm rmdir \
        vdir sleep stty sync touch true uname join"

for f in ${COREUTILS_BIN_PROGS} ; do \
        mv ${TARGET_DIR}/usr/bin/${f} ${TARGET_DIR}/bin \
done

ln -fs test ${TARGET_DIR}/usr/bin/[

mv ${TARGET_DIR}/usr/bin/chroot ${TARGET_DIR}/usr/sbin
```

coreutils - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

COREUTILS_BIN_PROGS="base64 cat chgrp chmod chown cp date dd df dir echo false \
        kill link ln ls mkdir mknod mktemp mv nice printenv pwd rm rmdir \
        vdir sleep stty sync touch true uname join"

for f in ${COREUTILS_BIN_PROGS} ; do \
        mv ${TARGET_DIR}/usr/bin/${f} ${TARGET_DIR}/bin \
done

ln -fs test ${TARGET_DIR}/usr/bin/[

mv ${TARGET_DIR}/usr/bin/chroot ${TARGET_DIR}/usr/sbin
```

采用传统 UNIX 的目录结构方式，即独立的 /bin、/sbin 和 /lib；而不是将它们合并到 /usr 下

coreutils - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

COREUTILS_BIN_PROGS="base64 cat chgrp chmod chown cp date dd df dir echo false \
        kill link ln ls mkdir mknod mktemp mv nice printenv pwd rm rmdir \
        vdir sleep stty sync touch true uname join"

for f in ${COREUTILS_BIN_PROGS} ; do \
        mv ${TARGET_DIR}/usr/bin/${f} ${TARGET_DIR}/b
done

ln -fs test ${TARGET_DIR}/usr/bin/[

mv ${TARGET_DIR}/usr/bin/chroot ${TARGET_DIR}/usr/
```

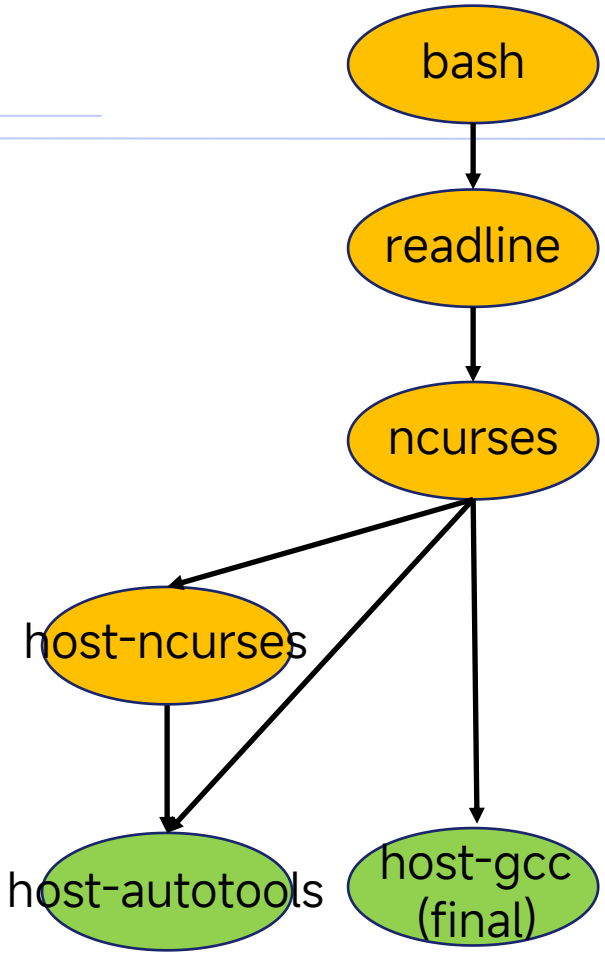
创建一个符号链接，将 test 命令链接到 /usr/bin/[这个特殊文件名。

```
test -f /etc/passwd
[ -f /etc/passwd ]
```

04

安装更好的 Shell（Bash）

Bash 制作过程


bash
readline
ncurses
host-ncurses
host-autotools
host-gcc
(final)

ncurses 简介

https://www.gnu.org/software/ncurses/

- ncurses 是一个用于在终端环境下创建文本用户界面的核心库。
- ncurses 最大的优势在于它提供了一套与终端类型无关的编程接口。无论用户使用的是哪种终端模拟器，程序都能正确地控制光标、绘制窗口和处理颜色，从而极大地简化了文本界面程序的开发。
- 日常使用的许多经典命令行工具在运行时都依赖于 ncurses，例如 vim、nano、tmux 等；Bash 在运行时所依赖的 readline 库在底层也依赖 ncurses 来获取控制终端的能力。
- 交叉构建 ncurses 库时需要先构建 host-ncurese，即在 ${HOST_DIR} 上安装一些工具譬如 tic 等。最主要的原因是因为交叉构建 ncurses 时需要在 Host 系统上运行这些工具来生成 Target 系统上的 terminfo 数据库。

host-ncurses – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} ac_cv_path_LDCONFIG=\"\"
CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --
sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --
disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-
documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-
dependency-tracking  --with-shared --without-gpm --without-manpages --without-cxx --
without-cxx-binding --without-ada --with-default-terminfo-dir=/usr/share/terminfo --disable-
db-install --without-normal"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装（install-host）

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

ncurses - 制作步驟

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
without-cxx --without-cxx-binding --without-ada --without-tests --disable-big-core --without-
profile --disable-rpath --disable-rpath-hack --enable-echo --enable-const --enable-overwrite -
-enable-pc-files --disable-stripping --with-pkg-config-libdir=\"/usr/lib/pkgconfig\" --without-
progs --without-manpages --with-shared --without-normal --without-gpm --disable-widec --
without-debug"
```

ncurses - 制作步驟

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
without-cxx --without-cxx-binding --without-ada --without-tests --disable-big-core --without-
profile --disable-rpath --disable-rpath-hack --enable-echo --enable-const --enable-overwrite -
-enable-pc-files --disable-stripping --with-pkg-config-libdir=\"/usr/lib/pkgconfig\" --without-
progs --without-manpages --with-shared --without-normal --without-gpm --disable-widec --
without-debug"
```

ncurses - 制作步驟

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j1  -C ${PKGBUILD_DIR}
DESTDIR=${STAGING_DIR} sources"

rm -rf ${PKGBUILD_DIR}/misc/pc-files

eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} -C ${PKGBUILD_DIR}
DESTDIR=${STAGING_DIR}"
```

ncurses - 制作步骤

安装 （ install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

rm -f -rf ${TARGET_DIR}/usr/share/terminfo ${TARGET_DIR}/usr/share/tabset

NCURSES_TERMINFO_FILES="a/ansi d/dumb l/linux p/putty p/putty-256color \
        p/putty-vt100 s/screen  s/screen-256color v/vt100 v/vt100-putty v/vt102 \
        v/vt200 v/vt220 x/xterm x/xterm+256color x/xterm-256color x/xterm-color \
        x/xterm-xfree86"

for f in ${NCURSES_TERMINFO_FILES} ; do
        /usr/bin/install -D -m 0644 ${STAGING_DIR}/usr/share/terminfo/${f}
                        ${TARGET_DIR}/usr/share/terminfo/${f}
done

rm -f -f ${TARGET_DIR}/usr/bin/ncurses6-config ;
```

ncurses - 制作步骤

安装 （ install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

rm -f -rf ${TARGET_DIR}/usr/share/terminfo ${TARGET_DIR}/usr/share/tabset

NCURSES_TERMINFO_FILES="a/ansi d/dumb l/linux p/putty p/putty-256color \
    p/putty-vt100 s/screen  s/screen-256color v/vt100 v/vt100-putty v/vt102 \
    v/vt200 v/vt220 x/xterm x/xterm+256color x/xterm-256color x/xterm-color \
    x/xterm-xfree86"

for f in ${NCURSES_TERMINFO_FILES} ; do
    /usr/bin/install -D -m 0644 ${STAGING_DIR}/usr/share/terminfo/${f}
                ${TARGET_DIR}/usr/share/terminfo/${f}
done

rm -f -f ${TARGET_DIR}/usr/bin/ncurses6-config ;
```

一种优化，显著缩减最终目标根文件系统中安装的 terminfo 数据库文件，只保留必要的终端描述信息，从而优化根文件系统的存储空间

readline 简介

https://www.gnu.org/software/readline/

- readline 软件库是一个提供命令行编辑和历史记录功能的 C 语言库。
- readline 软件库的核心功能包括：
  - ✓ 行编辑：支持光标移动（左/右箭头、Home/End）；插入和删除字符；基本的文本编辑操作
  - ✓ 历史记录：使用上下箭头浏览之前输入的命令；支持历史记录的搜索、保存和加载
  - ✓ 自动补全：按 Tab 键可以补全命令、文件名、变量名等，并可自定义补全逻辑。
  - ✓ 快捷键支持：提供了类似 Emacs 的默认快捷键（例如：Ctrl + A 跳到行首，Ctrl + E 跳到行尾，Ctrl + R 反向搜索历史等）；也可配置为 Vi 风格的快捷键模式
- Bash、zsh 等 Shell 程序都使用 readline 来提供强大的交互功能；此外还包括一些解释器环境（如 Python 等）以及需要复杂命令行交互的工具（如 GDB 等）。

readline - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  bash_cv_func_sigsetjmp=yes bash_cv_wcwidth_broken=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-install-examples --with-curses --with-shared-termcap-library --disable-bracketed-
paste-default"
```

readline - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  bash_cv_func_sigsetjmp=yes bash_cv_wcwidth_broken=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-install-examples --with-curses --with-shared-termcap-library --disable-bracketed-
paste-default"
```

readline - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （ install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} install -C
${PKGBUILD_DIR}"

/usr/bin/install -D -m 644 ${PROJECT_DIR}/package/readline/inputrc ${TARGET_DIR}/etc/inputrc
```

readline - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （ install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} install -C
${PKGBUILD_DIR}"

/usr/bin/install -D -m 644 ${PROJECT_DIR}/package/readline/inputrc ${TARGET_DIR}/etc/inputrc
```

bash 简介

https://www.gnu.org/software/bash/

- Bash（Bourne Again SHell）是 Linux 系统和 macOS（2019 年之前）默认的命令行解释器，也是目前最流行的 Unix shell 之一。它由 Brian Fox 于 1989 年为 GNU 项目开发，是 Bourne Shell（sh）的增强版。
- 查看当前 shell：echo $SHELL
- Linux 上常见的 Shell：

Shell 名称	开发时间	兼容性	脚本可移植性
Bash	1989	完全兼容 sh	高（接近 POSIX）
sh (Bourne Shell)	1977	标准 POSIX shell	最高（POSIX 标准）
zsh	1990	兼容 bash	中等

bash – 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  ac_cv_rl_prefix=\"${STAGING_DIR}\" ac_cv_rl_version=\"8.3\"
bash_cv_getcwd_malloc=yes bash_cv_job_control_missing=present
bash_cv_sys_named_pipes=present bash_cv_func_sigsetjmp=present
bash_cv_printf_a_format=yes CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu
--prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --
disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation
--with-xmlto=no --with-fop=no --disable-dependency-tracking --enable-ipv6 --disable-nls --
disable-static --enable-shared  --bindir=/bin --with-installed-readline --without-bash-malloc"
```

bash – 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  ac_cv_rl_prefix=\"${STAGING_DIR}\" ac_cv_rl_version=\"8.3\"
bash_cv_getcwd_malloc=yes bash_cv_job_control_missing=present
bash_cv_sys_named_pipes=present bash_cv_func_sigsetjmp=present
bash_cv_printf_a_format=yes CONFIG_SITE=/dev/null ./configure --
target=${GNU_TARGET_NAME} --host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu
--prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --
disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation
--with-xmlto=no --with-fop=no --disable-dependency-tracking --enable-ipv6 --disable-nls --
disable-static --enable-shared  --bindir=/bin --with-installed-readline --without-bash-malloc"
```

bash - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

rm -f ${TARGET_DIR}/bin/bashbug

rm -rf ${TARGET_DIR}/usr/lib/bash
```

bash – 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （ install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"

rm -f ${TARGET_DIR}/bin/bashbug

rm -rf ${TARGET_DIR}/usr/lib/bash
```

删除一些不需要的文件。

05

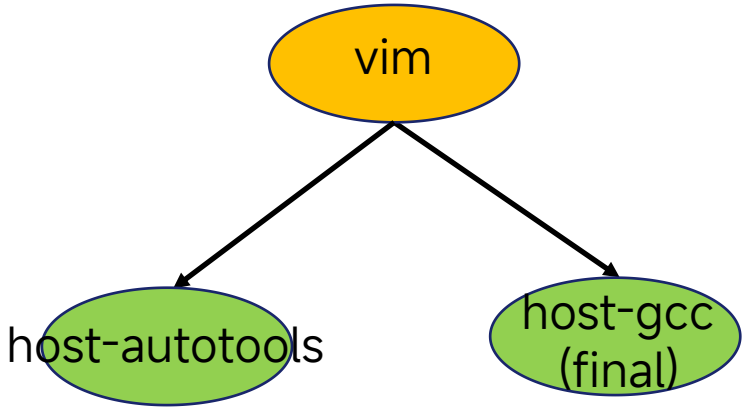
安装常用编辑软件（Vim）

Vim 简介

https://www.vim.org/

- Vim（ Vi IMproved ）是一个功能强大、高度可定制且高效的文本编辑器，是 Linux/Unix 系统管理员和开发者的经典工具。
- 完全兼容经典 Vi 编辑器，由 Bram Moolenaar 于 1991 年发布。
- 最重要的特性是模态编辑，包括普通模式、插入模式、可视模式和命令模式，支持全键盘操作： 减少手在键盘和鼠标间移动的损耗。
- 几乎所有的 Unix/Linux 系统和 macOS 都预装了 Vi/Vim，是远程服务器编辑的标配。

Vim 制作过程



vim - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  vim_cv_toupper_broken=no vim_cv_terminfo=yes
vim_cv_tgetent=zero vim_cv_tty_group=world vim_cv_tty_mode=0620
vim_cv_getcwd_broken=no vim_cv_stat_ignores_slash=yes
vim_cv_memmove_handles_overlap=yes ac_cv_sizeof_int=4 ac_cv_small_wchar_t=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
with-tlib=ncurses --enable-gui=no --without-x --disable-acl --disable-gpm --disable-selinux"
```

vim - 制作步骤

配置

```
eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\"
ac_cv_c_bigendian=no  vim_cv_toupper_broken=no vim_cv_terminfo=yes
vim_cv_tgetent=zero vim_cv_tty_group=world vim_cv_tty_mode=0620
vim_cv_getcwd_broken=no vim_cv_stat_ignores_slash=yes
vim_cv_memmove_handles_overlap=yes ac_cv_sizeof_int=4 ac_cv_small_wchar_t=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
with-tlib=ncurses --enable-gui=no --without-x --disable-acl --disable-gpm --disable-selinux"
```

vim – 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}/src"
```

安装 （ install-target)

```
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installvimbin"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installpack"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installtools"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installlinks"
ln -sf ../usr/bin/vim ${TARGET_DIR}/bin/vi
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installrtbase"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installmacros"
rm -f -rf ${TARGET_DIR}/usr/share/vim/vim*/doc/
```

vim - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}/src"
```

安装 （ install-target)

```
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installvimbin"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installpack"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installtools"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installlinks"
ln -sf ../usr/bin/vim ${TARGET_DIR}/bin/vi
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGE                    se"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGE                    ros"
rm -f -rf ${TARGET_DIR}/usr/share/vim/vim*/doc/
```

安装 Vim 的核心执行文件

vim - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}/src"
```

安装 （ install-target)

```
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installvimbin"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installpack"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installtools"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installlinks"
ln -sf ../usr/bin/vim ${TARGET_DIR}/bin/vi
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNU                                    tallrtbase"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNU                                    tallmacros"
rm -f -rf ${TARGET_DIR}/usr/share/vim/vim*/doc/
```

创建 vim 的符号链接 vi，兼容性考虑

vim - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}/src"
```

安装 （ install-target)

```
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installvimbin"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installpack"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} tools"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} links"
ln -sf ../usr/bin/vim ${TARGET_DIR}/bin/vi
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installrtbase"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installmacros"
rm -f -rf ${TARGET_DIR}/usr/share/vim/vim*/doc/
```

安装 Vim 的运行时支持文件

vim - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}/src"
```

安装 （ install-target)

```
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installvimbin"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installpack"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installtools"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installlinks"
ln -sf ../usr/bin/vim ${TARGET_DIR}/bin/vi
cd ${PKGBUILD_DIR}/src;
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installrtbase"
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} installmacros"
rm -f -rf ${TARGET_DIR}/usr/share/vim/vim*/doc/
```

删掉文档，缩减根文件系统体积

06

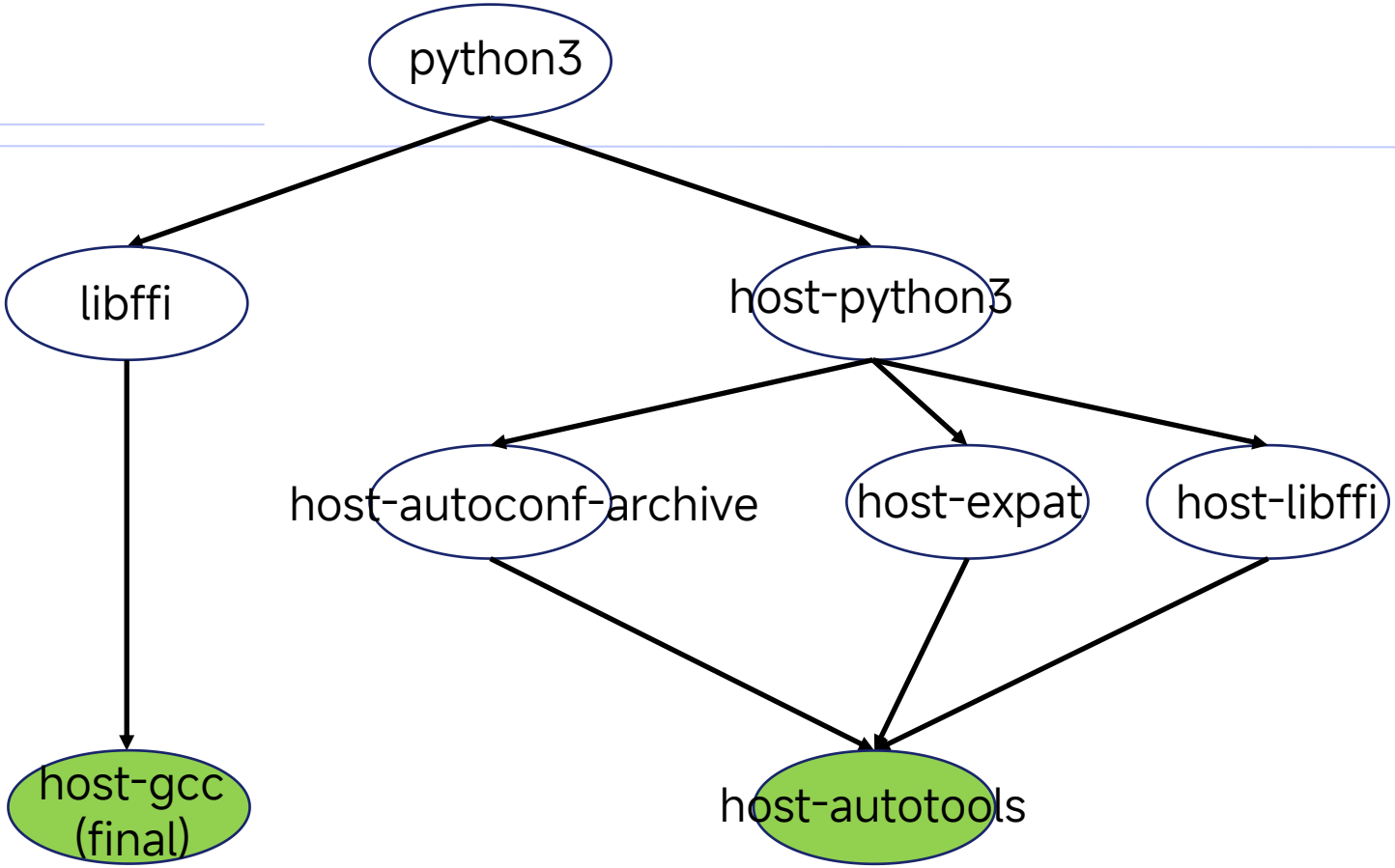
安装常用开发软件（Python）

Python 简介

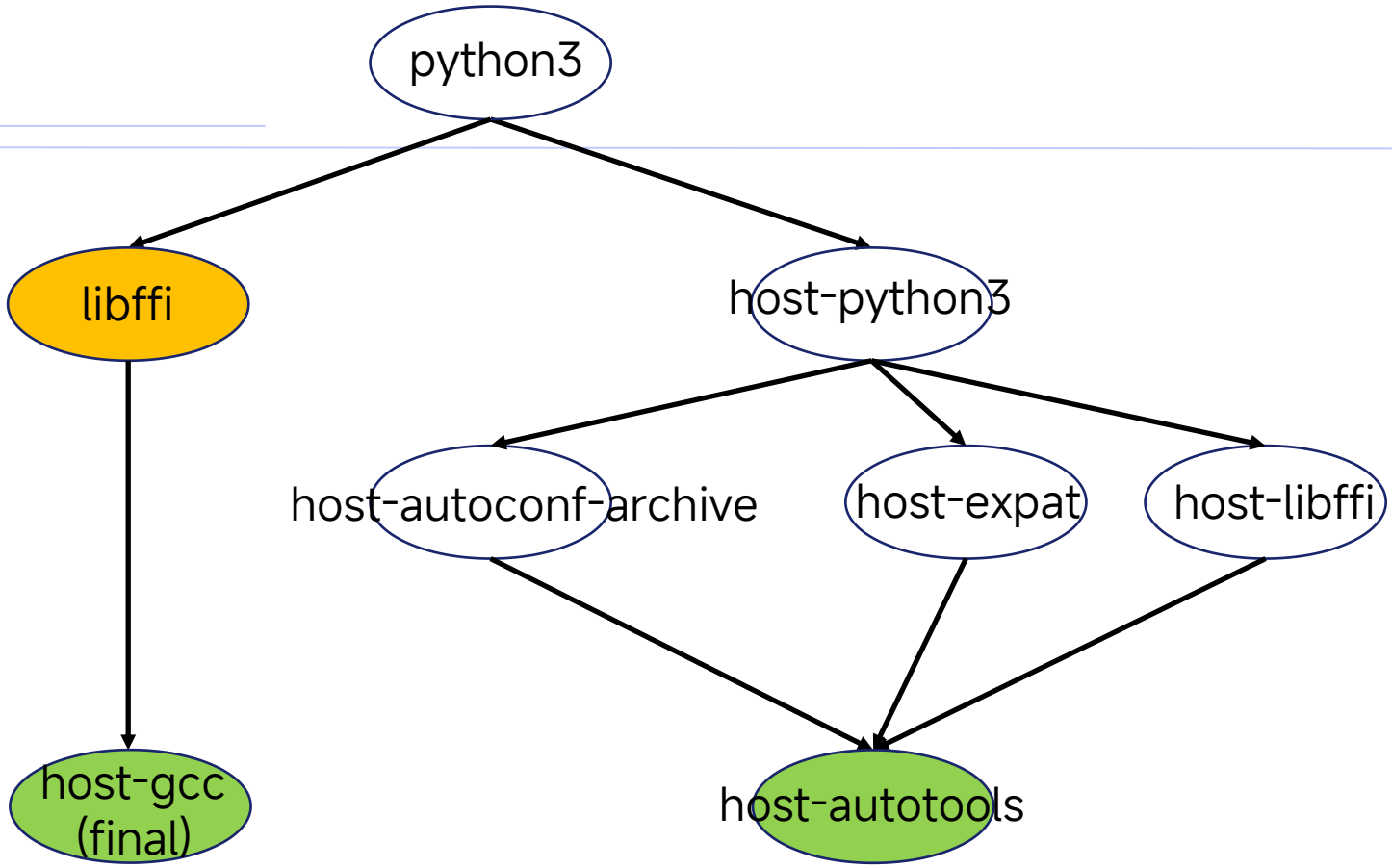
https://www.python.org/

- Python 是一种高级、解释型、通用编程语言，由 Guido van Rossum 于 1991年首次发布。它以其简洁易读的语法和强大的功能而广受欢迎。
- Python 的主要特点包括：
  - ✓ 简单易学
  - ✓ 解释型语言
  - ✓ 跨平台
  - ✓ 丰富的标准库和第三方库
- 在 Web开发、数据科学与机器学习、人工智能与深度学习、自动化与脚本和 游戏开发方面有着广泛的应用。
- Python 历史发展上分为 Python 1.x（1994-1999）；Python 2.x (2000-2010）；Python 3.x （2008 至今，不兼容 2.x）

python3 制作过程



python3 制作过程



Libffi 简介

https://sourceware.org/libffi/

- FFI （Foreign Function Interface）是一套允许一种编程语言安全、正确地调用另一种编程语言（尤其是 C 语言）编写的函数的机制。
- libffi 是一个 C 语言编写的开源库，作为 FFI 的一个实现，它提供一个统一的、高级的接口来屏蔽不同调用约定和底层机器的差异，从而实现在程序运行时正确地设置堆栈、传递参数，动态调用其他函数。
- Libffi 主要用于支持脚本语言的解释器使其能够调用 C/C++ 编写的库（如 Python 中使用 ctypes 或 cffi 库来调用一个 .so 文件中的 C 函数）。
- libffi 采用 MIT 许可证，在许多 Linux 发行版中作为基础库默认安装。

libffi - 制作步驟

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i"

patch_libtool ${PKGBUILD_DIR}

eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-multi-os-directory --disable-exec-static-tramp"
```

libffi - 制作步驟

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i"

patch_libtool ${PKGBUILD_DIR}

eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes
ac_cv_func_mmap_fixed_mapped=yes ac_cv_func_memcmp_working=yes
ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes
ac_cv_func_realloc_0_nonnull=yes lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no
CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --
sysconfdir=/etc --localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-
html --disable-doc --disable-docs --disable-documentation --with-xmlto=no --with-fop=no --
disable-dependency-tracking --enable-ipv6 --disable-nls --disable-static --enable-shared  --
disable-multi-os-directory --disable-exec-static-tramp"
```

libffi - 制作步骤

构建

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

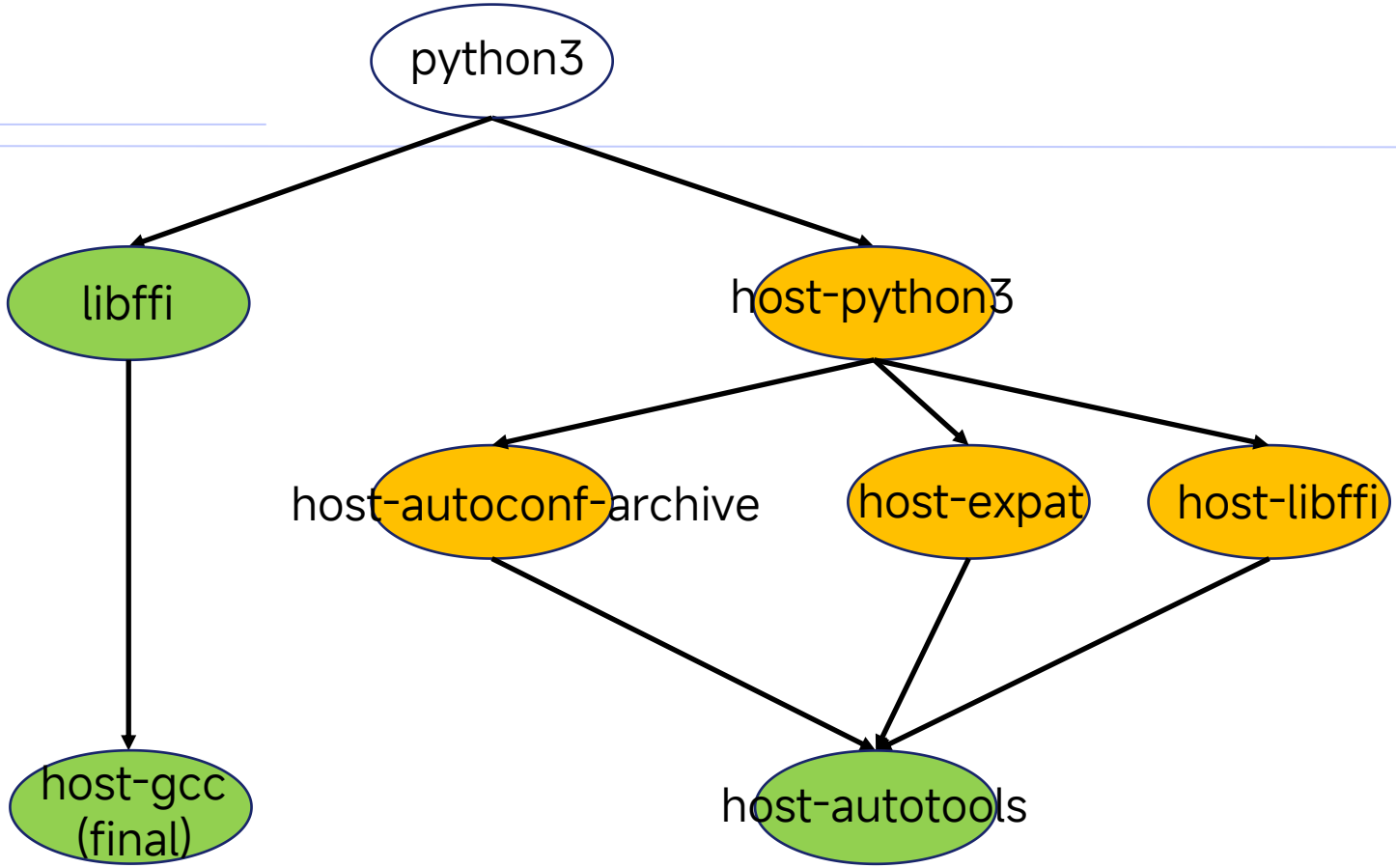
安装 （install-staging）

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （install-target）

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${TARGET_DIR} install -C ${PKGBUILD_DIR}"
```

python3 制作过程



Autoconf-archive 简介

https://www.gnu.org/software/autoconf-archive/

- Autoconf-archive 是 GNU 项目的一部分，它维护了有关 Autoconf 的宏集合，极大地扩展了 Autoconf 的功能，简化了复杂构建系统的配置过程。
- Autoconf-archive 提供丰富的预定义的可直接使用的 M4 宏，涵盖了常见的配置检查需求，避免开发者重复编写相同的配置代码。
- 构建 Python 的过程中，需要运行 autoreconf 来重新生成 configure 等文件。而 Python 的 configure.ac 使用了 autoconf-archive 提供的特定 M4 宏。

Autoconf-archive – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} aclocaldir=${HOST_DIR}/share/autoconf-archive install -C ${PKGBUILD_DIR}"
```

Expat 简介

https://libexpat.github.io/

- Expat（Expat XML parser）是一个用 C 语言编写的、开源且轻量的 XML 解析库。作为计算机软件中的一个基础组件，它的核心作用是为其他程序提供解析和处理 XML 数据的能力。
- Python 的某些核心库（特别是处理 XML 的库，譬如 pyexpat）依赖于底层的 expat 库来解析 XML。

Expat – 制作步骤

配置

```
eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking  --without-docbook --without-examples --without-tests"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C ${PKGBUILD_DIR}"
```

Host-Libffi – 制作步驟

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i"

patch_libtool ${PKGBUILD_DIR}

eval "${HOST_CONFIGURE_OPTS} CONFIG_SITE=/dev/null ./configure --
prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\"
--enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --
disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --
disable-nls --disable-dependency-tracking"
```

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"
```

Host-Python3 – 制作步驟

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i --include=${HOST_DIR}/share/autoconf-archive"

patch_libtool ${PKGBUILD_DIR}

eval "${HOST_CONFIGURE_OPTS} LDFLAGS=\"-L${HOST_DIR}/lib -Wl,-rpath,${HOST_DIR}/lib -Wl,--enable-new-dtags\" py_cv_module_unicodedata=yes py_cv_module__codecs_cn=n/a py_cv_module__codecs_hk=n/a py_cv_module__codecs_iso2022=n/a py_cv_module__codecs_jp=n/a py_cv_module__codecs_kr=n/a py_cv_module__codecs_tw=n/a py_cv_module__uuid=n/a py_cv_module_nis=n/a py_cv_module_ossaudiodev=n/a py_cv_module__bz2=n/a py_cv_module__lzma=n/a py_cv_module__hashlib=n/a py_cv_module__ssl=n/a CONFIG_SITE=/dev/null ./configure --prefix=\"${HOST_DIR}\" --sysconfdir=\"${HOST_DIR}/etc\" --localstatedir=\"${HOST_DIR}/var\" --enable-shared --disable-static --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --disable-documentation --disable-debug --with-xmlto=no --with-fop=no --disable-nls --disable-dependency-tracking  --without-ensurepip --without-cxx-main --disable-sqlite3 --disable-tk --with-expat=system --disable-test-modules --disable-idle3 --disable-curses"
```

Host-Python3 – 制作步骤

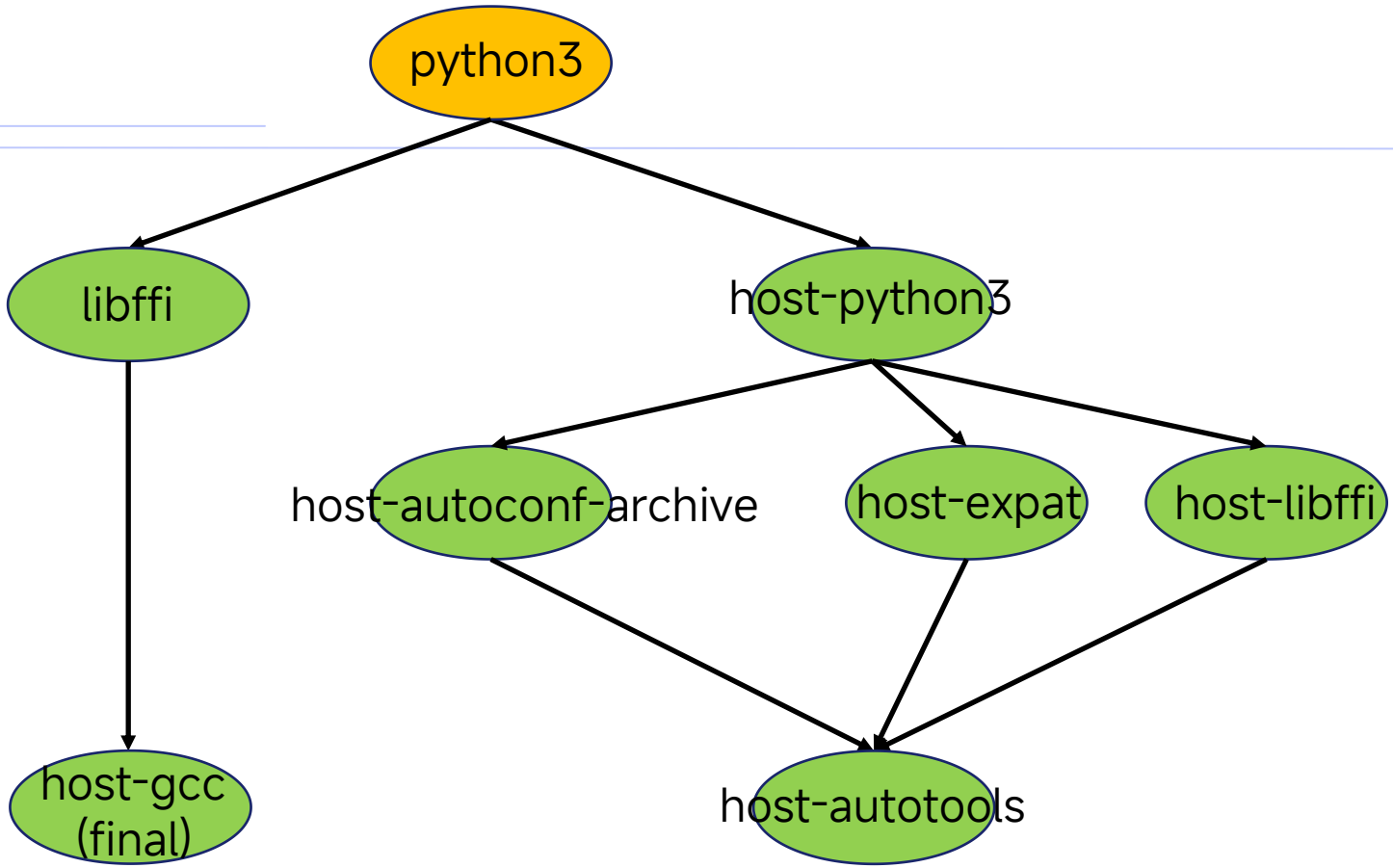
构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-host)

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} install -C
${PKGBUILD_DIR}"

ln -fs python3 ${HOST_DIR}/bin/python

ln -fs python3-config ${HOST_DIR}/bin/python-config
```

Python3 制作过程



Python3 - 制作步骤

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i --
include=${HOST_DIR}/share/autoconf-archive"

patch_libtool ${PKGBUILD_DIR}

eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes ac_cv_func_mmap_fixed_mapped=yes
ac_cv_func_memcmp_working=yes ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes ac_cv_func_realloc_0_nonnull=yes
lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no  py_cv_module__dbm=n/a
py_cv_module__decimal=n/a py_cv_module__hashlib=n/a py_cv_module__ssl=n/a py_cv_module__codecs_cn=n/a
py_cv_module__codecs_hk=n/a py_cv_module__codecs_iso2022=n/a py_cv_module__codecs_jp=n/a
py_cv_module__codecs_kr=n/a py_cv_module__codecs_tw=n/a py_cv_module__uuid=n/a
py_cv_module__bz2=n/a py_cv_module__lzma=n/a py_cv_module_zlib=n/a py_cv_module_ossaudiodev=n/a
ac_cv_have_long_long_format=yes ac_cv_buggy_getaddrinfo=no ac_cv_file__dev_ptmx=yes
ac_cv_file__dev_ptc=yes ac_cv_working_tzset=yes py_cv_module_nis=n/a ac_cv_little_endian_double=yes
CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -
D_FORTIFY_SOURCE=1\" CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --
localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --enable-ipv6 --disable-
nls --disable-static --enable-shared  --disable-lib2to3 --without-readline --with-expat=none --disable-sqlite3 --
without-ensurepip --without-cxx-main --with-build-python=${HOST_DIR}/bin/python3 --with-system-ffi --
disable-pydoc --disable-test-modules --disable-tk --disable-idle3 --disable-pyc-build"
```

Python3 - 制作步骤

配置

```
eval "AUTOPOINT=/bin/true ${AUTORECONF_OPTS} ${HOST_DIR}/bin/autoreconf -f -i --
include=${HOST_DIR}/share/autoconf-archive"

patch_libtool ${PKGBUILD_DIR}

eval "${TARGET_CONFIGURE_OPTS} CXX=no ac_cv_lbl_unaligned_fail=yes ac_cv_func_mmap_fixed_mapped=yes
ac_cv_func_memcmp_working=yes ac_cv_have_decl_malloc=yes gl_cv_func_malloc_0_nonnull=yes
ac_cv_func_malloc_0_nonnull=yes ac_cv_func_calloc_0_nonnull=yes ac_cv_func_realloc_0_nonnull=yes
lt_cv_sys_lib_search_path_spec=\"\" ac_cv_c_bigendian=no  py_cv_module__dbm=n/a
py_cv_module__decimal=n/a py_cv_module__hashlib=n/a py_cv_module__ssl=n/a py_cv_module__codecs_cn=n/a
py_cv_module__codecs_hk=n/a py_cv_module__codecs_iso2022=n/a py_cv_module__codecs_jp=n/a
py_cv_module__codecs_kr=n/a py_cv_module__codecs_tw=n/a py_cv_module__uuid=n/a
py_cv_module__bz2=n/a py_cv_module__lzma=n/a py_cv_module_zlib=n/a py_cv_module_ossaudiodev=n/a
ac_cv_have_long_long_format=yes ac_cv_buggy_getaddrinfo=no ac_cv_file__dev_ptmx=yes
ac_cv_file__dev_ptc=yes ac_cv_working_tzset=yes py_cv_module_nis=n/a ac_cv_little_endian_double=yes
CFLAGS=\"-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64  -O2 -g0 -
D_FORTIFY_SOURCE=1\" CONFIG_SITE=/dev/null ./configure --target=${GNU_TARGET_NAME} --
host=${GNU_TARGET_NAME} --build=x86_64-pc-linux-gnu --prefix=/usr --exec-prefix=/usr --sysconfdir=/etc --
localstatedir=/var --program-prefix=\"\" --disable-gtk-doc --disable-gtk-doc-html --disable-doc --disable-docs --
disable-documentation --with-xmlto=no --with-fop=no --disable-dependency-tracking --enable-ipv6 --disable-
nls --disable-static --enable-shared  --disable-lib2to3 --without-readline --with-expat=none --disable-sqlite3 --
without-ensurepip --without-cxx-main --with-build-python=${HOST_DIR}/bin/python3 --with-system-ffi --
disable-pydoc --disable-test-modules --disable-tk --disable-idle3 --disable-pyc-build"
```

Python3 - 制作步骤

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-staging)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （install-target)

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} install -C
${PKGBUILD_DIR}"

rm -f ${TARGET_DIR}/usr/bin/python3.13-config

rm -f ${TARGET_DIR}/usr/bin/python3-config

find ${TARGET_DIR}/usr/lib/python3.13/config-3.13*/ -depth -type f -not -name Makefile -exec rm -rf {} \;

find ${TARGET_DIR}/usr/lib/python3.13/ -depth -type d -name __pycache__ -exec rm -rf {} \;

chmod u+w ${TARGET_DIR}/usr/lib/libpython3.13*.so

ln -fs python3 ${TARGET_DIR}/usr/bin/python
```

Python3 – 制作步骤

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-staging）

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （install-target）

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} install -C
${PKGBUILD_DIR}"
rm -f ${TARGET_DIR}/usr/bin/python3.13-config
rm -f ${TARGET_DIR}/usr/bin/python3-config
find ${TARGET_DIR}/usr/lib/python3.13/config-3.13*/ -depth -type f -not -name Makefile -exec rm -rf {} \;
find ${TARGET_DIR}/usr/lib/python3.13/ -depth -type d -name __pycache__ -exec rm -rf {} \;
chmod u+w ${TARGET_DIR}/usr/lib/libpython3.13*.so
ln -fs python3 ${TARGET_DIR}/usr/bin/python
```

删除一些 runtime 期间不需要的文件

Python3 – 制作步骤

构建

```
eval "${HOST_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}  -C ${PKGBUILD_DIR}"
```

安装 （install-staging）

```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS}
DESTDIR=${STAGING_DIR} install -C ${PKGBUILD_DIR}"
```

安装 （install-target）

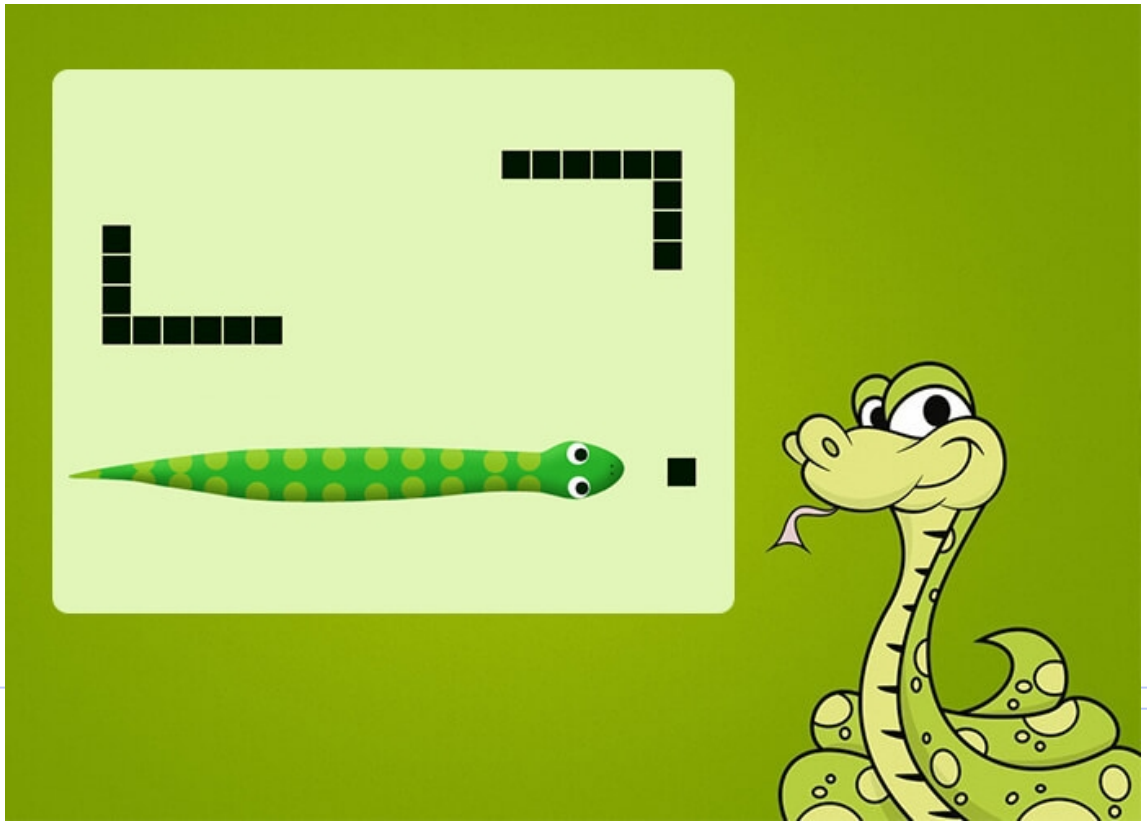
```
eval "${TARGET_MAKE_ENV} /usr/bin/make -j${MAXNUM_CPUS} DESTDIR=${TARGET_DIR} install -C
${PKGBUILD_DIR}"

rm -f ${TARGET_DIR}/usr/bin/python3.13-config

rm -f ${TARGET_DIR}/usr/bin/python3-config

find ${TARGET_DIR}/usr/lib/python3.13/config-3.13*/ -depth -t                    rm -rf {} \;

find ${TARGET_DIR}/usr/lib/python3.13/ -depth -type d -nam

chmod u+w ${TARGET_DIR}/usr/lib/libpython3.13*.so

ln -fs python3 ${TARGET_DIR}/usr/bin/python
```

确保后期可以删除 libpython 库文件中的调试符号以缩减根文件系统体积。

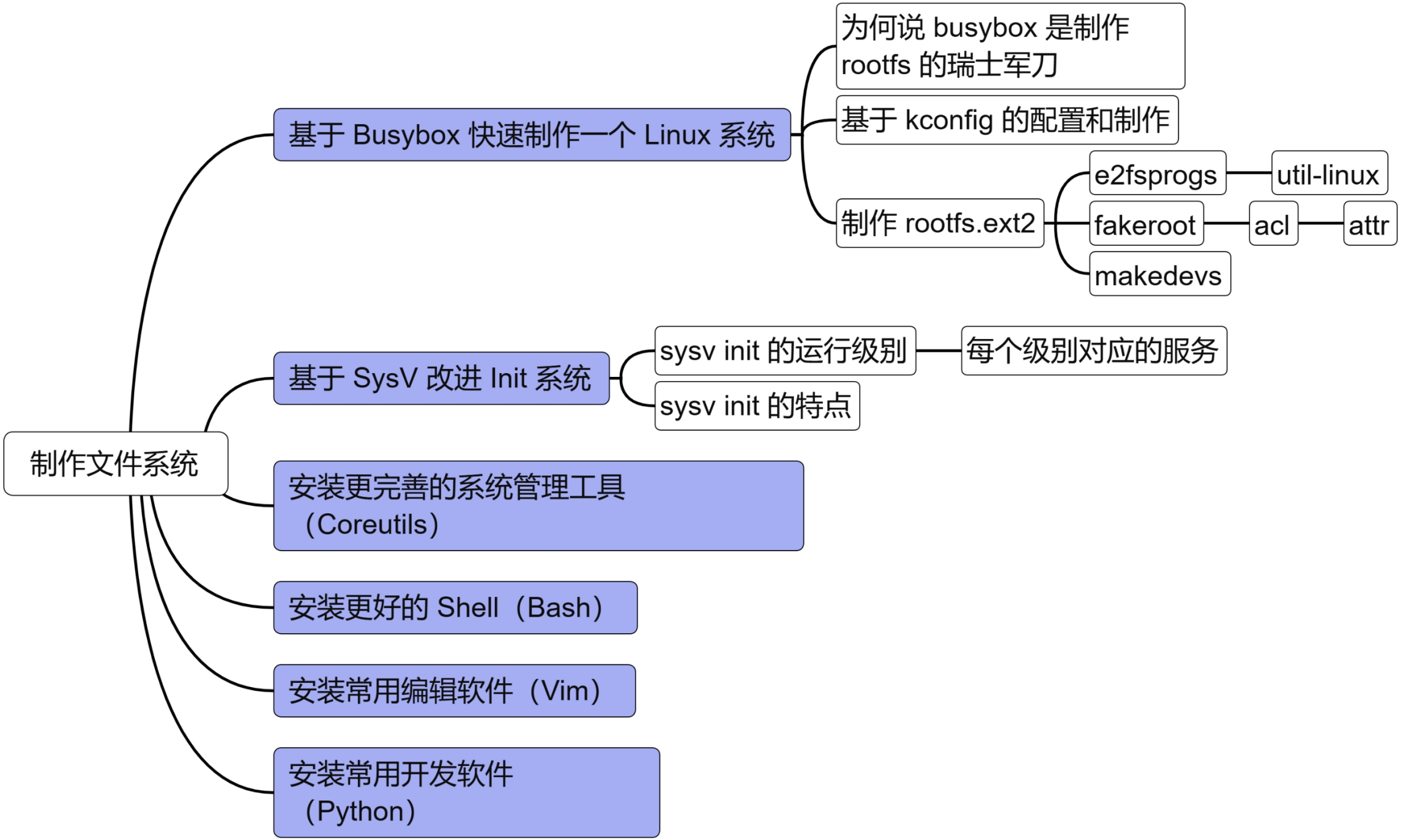
安装一个 Python 小游戏

安装 （install-target)

```
mkdir -p ${TARGET_DIR}/usr/games
cp ${PROJECT_DIR}/package/${PKGNAME}/snake.py ${TARGET_DIR}/usr/games/
```



本章总结


制作文件系统
基于 Busybox 快速制作一个 Linux 系统
为何说 busybox 是制作 rootfs 的瑞士军刀
基于 kconfig 的配置和制作
制作 rootfs.ext2
e2fsprogs
util-linux
fakeroot
acl
attr
makedevs
基于 SysV 改进 Init 系统
sysv init 的运行级别
每个级别对应的服务
sysv init 的特点
安装更完善的系统管理工具（Coreutils）
安装更好的 Shell（Bash）
安装常用编辑软件（Vim）
安装常用开发软件（Python）

谢谢