# 基于 deepin 的操作系统构建

## 第 11 课　构建 Linux 操作系统交叉工具链和临时工具 - 进入 chroot 构建

徐小东

统信软件

2022-10-31

# 课程目录

Section 1

# 进入 chroot 构建

# 进入 chroot 构建

- chroot
  针对当前运行的进程及其子进程更改根目录的操作
- 应用
  创建与宿主机隔离的虚拟系统环境
- 在 chroot 中构建临时系统最后缺失的部分

Section 2

# 需要的软件包

# 需要的软件包

- 总共 7 个

| 包名 | 版本 | 用途 |
| --- | --- | --- |
| GCC | 11.2.0 | C、C++ 编译器 |
| Gettext | 0.21 | 国际化和本地化工具 |
| Bison | 3.8.2 | 语法分析生成器 |
| Perl | 5.34.0 | 编程语言 |
| Python | 3.10.2 | 编程语言 |
| Texinfo | 6.8 | info 页面程序 |
| Util-linux | 2.37.4 | 各种工具程序 |

Section 3

# 构建缺少的临时工具

# 改变目录所有者

```
# 切换到 root
sudo -i

# 设置 $LFS 环境变量
export LFS=/home/deepin/lfs

# 将 $LFS/* 目录的所有者改变为 root
chown -R root:root $LFS/{usr,lib,lib64,var,
  etc,bin,sbin,tools}
```

# 准备虚拟内核文件系统

```
# 创建这些文件系统的挂载点
mkdir -pv $LFS/{dev,proc,sys,run}
# 创建初始设备节点
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3

mount -v --bind /dev $LFS/dev
mount -v --bind /dev/pts $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run

if [ -h $LFS/dev/shm ]; then
  mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

# 进入 Chroot 环境

```
# 进入 chroot 环境并完成剩余临时工具的安装
chroot "$LFS" /usr/bin/env -i   \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/usr/bin:/usr/sbin \
    /bin/bash --login +h
```

# 创建完整的目录结构

- 在 LFS 文件系统中创建完整的目录结构，参考 FHS[1]

```
mkdir -pv /{boot,home,mnt,opt,srv}
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,
  locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}
```

[1] https://refspecs.linuxfoundation.org/fhs.shtml

```
ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

# 创建必要的文件和符号链接

```
# 已经挂载的文件系统的列表
ln -sv /proc/self/mounts /etc/mtab

# 创建一个基本的 /etc/hosts 文件
cat > /etc/hosts << EOF
127.0.0.1  localhost lfs
::1        localhost
EOF
```

```
# 创建默认的用户
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:
  /bin/false
systemd-bus-proxy:x:72:72:systemd Bus Proxy:/:/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:
  /bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:
  /bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:
  /bin/false
```

```
systemd-network:x:76:76:systemd Network Management:/:
  /bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:
  /bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:
  /bin/false
systemd-oom:x:81:81:systemd Out Of Memory Daemon:/:
  /bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

```
# 创建默认的组
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
```

```
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
systemd-bus-proxy:x:72:
systemd-journal-gateway:x:73:
```

```
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
uuidd:x:80:
systemd-oom:x:81:81:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

```
# 为 login、agetty 和 init 初始化日志文件
touch /var/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

# 构建 GCC 中的 Libstdc++

- 避免编译产生的库被宿主系统组件污染

```
# 允许在 GCC 源码树中构建 Libstdc++
ln -s gthr-posix.h libgcc/gthr-default.h

mkdir -v build
cd       build
```

```
../libstdc++-v3/configure                    \
    CXXFLAGS="-g -O2 -D_GNU_SOURCE"   \ (1)
    --prefix=/usr                            \
    --host=$(uname -m)-lfs-linux-gnu \
    --disable-multilib                       \
    --disable-nls                            \
    --disable-libstdcxx-pch (2)

make
make install
```

(1). 通过顶层 Makefile 传递编译选项

(2). 阻止安装预编译的 include 文件，此阶段不需要

# 构建 Gettext

- 包含国际化和本地化工具

```
./configure --disable-shared (1)

make -j1

cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} \
  /usr/bin
```

(1). 不构建共享库，此阶段不需要

# 构建 Bison

- 包含语法分析器生成器

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2 (1)

make
make install
```

(1). 将文档安装到版本化的目录

# 构建 Perl

## 编程语言

```
sh Configure -des \ (1)
             -Dprefix=/usr \
             -Dvendorprefix=/usr \
             -Dprivlib=/usr/lib/perl5/5.34/core_perl \
             -Darchlib=/usr/lib/perl5/5.34/core_perl \
             -Dsitelib=/usr/lib/perl5/5.34/site_perl \
             -Dsitearch=/usr/lib/perl5/5.34/site_perl \
             -Dvendorlib=/usr/lib/perl5/5.34/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.34/vendor_perl

make
make install
```

(1). -d 使用默认值，-e 确保编译所有任务，-s 静默非重要输出

# 构建 Python

- 编程语言

```
./configure --prefix=/usr    \
            --enable-shared \ (1)
            --without-ensurepip (2)

make
make install
```

(1). 防止安装静态库

(2). 禁止构建 Python 软件包安装器，它在当前阶段没有必要

# 构建 Texinfo

- 包含阅读、编写和转换 info 页面的程序

```
# 修复构建错误
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c

./configure --prefix=/usr

make
make install
```

# 构建 Util-linux

- 各种工具程序

```
# 作为 adjtime 文件的位置
mkdir -pv /var/lib/hwclock

./configure ADJTIME_PATH=/var/lib/hwclock/adjtime \ (1)
            --libdir=/usr/lib    \ (2)
            --docdir=/usr/share/doc/util-linux-2.37.4 \
            --disable-chfn-chsh  \ (3)
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
```

```
            --disable-runuser     \
            --disable-pylibmount \
            --disable-static      \ (4)
            --without-python      \ (5)
            --runstatedir=/run (6)

make
make install
```

(1). 设置硬件时钟记录信息的位置

(2). 确保 .so 的符号链接与共享库为同一目录

(3)~(4). 阻止因尚不存在的包所引起的警告

(5). 禁用 Python 支持，避免构建不需要的绑定

(6). 设置 socket 的位置

# 清理和备份临时系统

```
# 删除临时工具的文档
rm -rf /usr/share/{info,man,doc}/*

# libtool .la 文件仅在链接到静态库时有用
find /usr/{lib,libexec} -name \*.la -delete

# 已经不需要其中的 /tools 目录
rm -rf /tools

# 离开 chroot 环境
exit
```

```
# 可以将临时系统备份起来，以便以后重新使用
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}

cd $LFS
tar cJpf /home/deepin/lfs-rootfs-11.1.tar.xz .

# 若需要时还原临时系统
cd $LFS
rm -rf ./*
tar xpf /home/deepin/lfs-rootfs-11.1.tar.xz
```

Section 4

# 补充阅读材料

# 补充阅读材料

- 《精通 Linux》
  - 第 3.1 节　设备文件
  - 第 7.3 节　用户管理文件

Section 5

# 参考资料

# 参考资料

- chroot
- hosts
- passwd
- group
- GCC Option Summary
- gettext
- GNU Bison
- Perl
- Python
- GNU Texinfo
- util-linux