7. Вход в окружение Chroot и создание дополнительных временных инструментов

Содержание

7.1. Введение

В этой главе рассказывается, как собрать последние недостающие части временной системы: инструменты, необходимые для сборки различных пакетов. Теперь, когда все циклические зависимости устранены, для сборки можно использовать среду «chroot», полностью изолированную от операционной системы хоста (за исключением работающего ядра).

Для правильной работы изолированной среды необходимо установить связь с работающим ядром. Это делается с помощью так называемых виртуальных файловых систем ядра, которые будут смонтированы перед входом в среду chroot. Вы можете проверить, смонтированы ли они, выполнив команду findmnt.

До Раздела 7.4. «Вход в окружение Chroot» команды должны выполняться от имени root с установленной переменной LFS. После входа в chroot все команды выполняются от имени root, к счастью, без доступа к операционной системе компьютера, на котором вы собираете LFS. В любом случае будьте осторожны, так как неверными командами легко разрушить всю систему LFS.

7.2. Смена владельца

Примечание
Команды, приведенные в оставшейся части книги, должны выполняться от имени пользователя root, а не lfs. Дважды проверьте, что переменная $LFS установлена в переменных окружения пользователя root.

В настоящее время вся иерархия каталогов в $LFS принадлежит пользователю **lfs**, существующему только на хост-системе. Если права на файлы и каталоги внутри $LFS оставить как есть, то они будут принадлежать ID пользователя без существующей учетной записи. Это опасно, так как созданная позже учетная запись, может получить такой же ID пользователя и стать владельцем всех файлов в $LFS, тем самым делая эти файлы уязвимыми для возможных злонамеренных манипуляций.

Для решения проблемы измените владельца каталогов $LFS/* на пользователя **root**, выполнив следующую команду:

```
chown -R root:root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown -R root:root $LFS/lib64 ;;
esac
```

7.3. Подготовка виртуальных файловых систем ядра

Приложения, работающие в пользовательском пространстве, используют различные файловые системы, созданные ядром, для взаимодействия с самим ядром. Эти файловые системы являются виртуальными: для них не используется дисковое пространство. Содержимое файловых систем находится в памяти. Эти файловые системы должны быть смонтированы в дереве каталогов $LFS, чтобы приложения могли найти их в среде **chroot**.

Начните с создания каталогов, в которые будут смонтированы эти виртуальные файловые системы:

```
mkdir -pv $LFS/{dev,proc,sys,run}
```

7.3.1. Монтирование и заполнение /dev

Во время обычной загрузки ядро автоматически монтирует файловую систему devtmpfs в каталог /dev; ядро создает узлы устройств в этой виртуальной файловой системе в процессе загрузки или при первом обнаружении устройства, или доступе к нему. Демон udev может изменять владельца или разрешения узлов устройств, созданных ядром, или создавать новые узлы устройств или символические ссылки, чтобы облегчить работу разработчиков дистрибутива или системных администраторов. (Подробности смотрите в Разделе 9.3.2.2. «Создание узла устройства».) Если ядро хоста поддерживает devtmpfs, мы можем просто смонтировать devtmpfs в $LFS/dev и положиться на ядро для его заполнения.

Но в некоторых ядрах хоста отсутствует поддержка devtmpfs, эти хост-дистрибутивы используют разные методы для создания содержимого /dev. Таким образом, единственный независимый от хоста способ заполнить каталог $LFS/dev - это привязка к каталогу /dev хост-системы. Связное монтирование - это особый тип монтирования, который делает дерево каталога или файл видимым в каком-либо другом месте. Для этого используйте следующую команду:

```
mount -v --bind /dev $LFS/dev
```

7.3.2. Монтирование виртуальных файловых систем ядра

Теперь смонтируйте оставшиеся виртуальные файловые системы:

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Значение параметров монтирования для devpts:

- **gid=5**

Этот параметр гарантирует, что все узлы устройств, созданные devpts, принадлежат группе с идентификатором 5. Это идентификатор, который мы будем использовать позже для группы tty. Мы используем идентификатор группы вместо имени, поскольку хост-система может использовать другой идентификатор для своей группы tty.

- **mode=0620**

Этот параметр гарантирует, что все узлы устройств, созданные devpts, будут иметь права 0620 (доступен для чтения и записи пользователем, доступен для записи группе). Вместе с вышеуказанной опцией это гарантирует, что devpts создаст узлы устройств, соответствующие требованиям grantpt(), а это означает, что вспомогательный файл Glibc pt_chown (который не установлен по умолчанию) не требуется.

В некоторых хост-системах /dev/shm является символической ссылкой на каталог /run/shm. /run tmpfs был смонтирован выше, поэтому сейчас необходимо только создать каталог с правильными разрешениями.

В других хост-системах /dev/shm является точкой монтирования для tmpfs. В этом случае монтирование /dev приведет только к созданию /dev/shm как каталога в среде chroot. В этой ситуации мы должны явно смонтировать tmpfs:

```
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. Вход в окружение Chroot

Теперь, когда все пакеты, необходимые для сборки остальных инструментов установлены в системе, пришло время войти в окружение chroot и завершить установку временных инструментов. Эта среда также будет использоваться для установки конечной системы. От имени пользователя root выполните следующую команду для входа в chroot, в которой на данный момент нет ничего, кроме временных инструментов:

```
chroot "$LFS" /usr/bin/env -i   \
```

```
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/usr/bin:/usr/sbin     \
    MAKEFLAGS="-j$(nproc)"      \
    TESTSUITEFLAGS="-j$(nproc)" \
    /bin/bash --login
```

Если вы не хотите использовать все доступные логические ядра, замените параметр $(nproc) количеством логических ядер, которые вы хотите использовать для сборки пакетов в этой и последующих главах. На наборы тестов некоторых пакетов (в частности Autoconf, Libtool и Tar) в Главе 8 не влияет установка переменной MAKEFLAGS, вместо этого они используют переменную среды TESTSUITEFLAGS. Мы также установили её здесь для запуска тестов с поддержкой нескольких ядер.

Параметр -i команды env, очистит все переменные в среде chroot. После этого переменные HOME, TERM, PS1 и PATH будут установлены заново. Конструкция TERM=$TERM установит переменную TERM внутри chroot в то же значение, что и вне chroot. Эта переменная необходима для корректной работы таких программ как vim и less. Если понадобятся другие переменные окружения, такие как CFLAGS или CXXFLAGS, то это подходящее место для их установки.

С этого момента больше нет необходимости использовать переменную LFS, поскольку вся работа будет ограничена файловой системой LFS; команда chroot запускает оболочку Bash с корневым каталогом (/), установленным в $LFS.

Обратите внимание, что каталог **/tools/bin** не указан в переменной окружения PATH. Это означает, что кросс-тулчейн больше не будет использоваться.

Также обратите внимание, что в командной строке bash будет указано I have no name!. Это нормально, поскольку файл **/etc/passwd** еще не создан.

Примечание
Важно, чтобы все команды в оставшейся части этой главы и следующих главах выполнялись из среды chroot. Если вы покидаете эту среду по какой-либо причине (например, при перезагрузке), убедитесь, что файловые системы виртуального ядра смонтированы, как описано в Разделе 7.3.1. «Монтирование и заполнение /dev» и Разделе 7.3.2. «Монтирование виртуальных файловых систем ядра», а затем войдите в среду chroot для продолжения установки.

7.5. Создание каталогов

Пришло время создать полную структуру каталогов в файловой системе LFS.

Примечание
Некоторые из каталогов, упомянутых в этом разделе, возможно, уже были созданы ранее с помощью явных инструкций или при установке некоторых пакетов. Они повторяются ниже для полноты картины.

Создайте несколько каталогов, которые не входили в ограниченный набор, используемый в предыдущих главах, выполнив следующую команду:

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Создайте необходимые подкаталоги, выполнив следующие команды:

```
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}

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

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

По умолчанию каталоги создаются с правами 755, но это нежелательно делать для всех каталогов. В приведенных выше командах вносятся два изменения — одно в домашний каталог пользователя root, а другое в каталоги для временных файлов.

Первое изменение гарантирует, что никто не сможет войти в каталог /root — точно так же, как обычный пользователь сделал бы это со своим собственным домашним каталогом. Второе изменение гарантирует, что любой пользователь может писать в каталоги /tmp и /var/tmp, но не может удалять из них файлы другого пользователя. Последнее запрещено так называемым «sticky bit (липким битом)», старшим битом (1) в битовой маске 1777

7.5.1. Примечание о соответствии требованиям FHS

Это дерево каталогов основано на стандарте иерархии файловой системы (FHS) (доступен по адресу https://refspecs.linuxfoundation.org/fhs.shtml). FHS также указывает, что наличие некоторых каталогов необязательно, например, /usr/local/games и /usr/share/games. В LFS мы создаем только те каталоги, которые действительно необходимы. Однако, не стесняйтесь создавать дополнительные каталоги, если хотите.

Предупреждение
FHS не требует наличия каталога /usr/lib64, и редакторы LFS решили его не использовать. Чтобы инструкции в LFS и BLFS работали корректно, крайне важно, чтобы этот каталог не существовал. Время от времени вам следует проверять, что он не существует, потому что его легко создать непреднамеренно, и это, вероятно, приведет к поломке вашей системы.

7.6. Создание основных файлов и символических ссылок

Исторически сложилось, что Linux хранит список примонтированных файловых систем в файле **/etc/mtab**. Современные ядра хранят этот список внутри себя и предоставляют его пользователю через файловую систему **/proc**. Чтобы удовлетворять требованиям утилит, которые ожидают наличия **/etc/mtab**, создайте следующую символическую ссылку:

```
ln -sv /proc/self/mounts /etc/mtab
```

Создайте файл **/etc/hosts**, на который будут ссылаться некоторые наборы тестов, а также один из файлов конфигурации Perl:

```
cat > /etc/hosts << EOF
127.0.0.1  localhost $(hostname)
::1        localhost
EOF
```

Чтобы пользователь **root** мог войти в систему и распознавался системой, в файлах **/etc/passwd** и **/etc/group** должны быть соответствующие записи.

Создайте файл **/etc/passwd** выполнив следующую команду:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/usr/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/usr/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/usr/bin/false
systemd-network:x:76:76:systemd Network Management:/:/usr/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/usr/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/usr/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
systemd-oom:x:81:81:systemd Out Of Memory Daemon:/:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Пароль пользователя **root** будет задан позднее.

Создайте файл **/etc/group**, выполнив следующую команду:

```
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:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
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:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Созданные группы не являются частью какого-либо стандарта — это группы, определяемые частично требованиями конфигурации Udev в Главе 9, а частично общими соглашениями, используемыми в ряде существующих дистрибутивов Linux. Кроме того, некоторые наборы тестов зависят от конкретных пользователей или групп. Спецификация LSB (доступна по адресу https://refspecs.linuxfoundation.org/lsb.shtml) рекомендует, чтобы, помимо группы **root** с идентификатор (GID) 0 присутствовала группа bin с GID 1. GID 5 широко используется для группы tty, число 5 также используется в systemd для файловой системы **devpts**. Все остальные имена групп и GID могут свободно выбираться системным администратором, так как хорошо написанные программы не зависят от номеров GID, а чаще используют название группы.

Идентификатор 65534 используется ядром для NFS и отдельных пользовательских пространств имен для несопоставленных пользователей и групп (они существуют на сервере NFS или родительском пространстве имен пользователя, но «не существует» на локальном компьютере или в отдельном пространстве имен). Мы присваиваем **nobody** и **nogroup** для того, чтобы избежать несопоставленных идентификаторов. Другие дистрибутивы могут обрабатывать этот идентификатор по-разному, поэтому любая переносимая программа не должна зависеть от этого присвоения.

Для некоторых тестов в Главе 8 требуется обычный пользователь. Добавим такого пользователя здесь и удалим эту учетную запись в конце главы.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Чтобы удалить приглашение «I have no name!», запустите новую оболочку. Поскольку файлы **/etc/passwd** и **/etc/group** были созданы, разрешение имен пользователей и групп теперь будет работать:

```
exec /usr/bin/bash --login
```

Программы **login, agetty, init (и другие)** используют ряд журналов для записи такой информации, как кто и когда входил в систему. Однако эти программы не будут записывать данные в журналы, если они еще не существуют. Инициализируйте журналы и предоставьте им соответствующие разрешения:

```
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
```

В файл **/var/log/wtmp** записываются все входы и выходы из системы. В файл **/var/log/lastlog** записывается время последнего входа каждого пользователя в систему. В файл **/var/log/faillog** записываются неудачные попытки входа в систему. В файл **/var/log/btmp** также записываются неудачные попытки входа в систему.

Примечание
Файлы wtmp, btmp и lastlog используют для временных меток 32-разрядные целые числа, значения счетчика достигнет максимума (2 147 483 647) 19 января 2038 года («проблема 2038 года»). Многие пакеты перестали их использовать, другие же, собираются прекратить их использование. Вероятно, лучше считать их устаревшими.

7.7. Gettext-0.22.4

Пакет Gettext содержит утилиты для интернационализации и локализации. Они позволяют компилировать программы с поддержкой NLS (Native Language Support), позволяя им выводить сообщения на родном языке пользователя.	
Приблизительное время сборки:	1.1 SBU
Требуемое дисковое пространство:	306 MB

7.7.1. Установка пакета Gettext

Для временного набора инструментов нам нужно установить только три программы из пакета Gettext.

Подготовьте Gettext к компиляции:

```
./configure --disable-shared
```

Значение параметров настройки:

- **–disable-shared**

В настоящее время нам не нужно устанавливать какие-либо общие библиотеки Gettext, поэтому нет необходимости их собирать.

Скомпилируйте пакет:

```
make
```

Установите программы msgfmt, msgmerge, и xgettext programs:

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

Подробная информация об этом пакете находится в Разделе 8.32.2. «Содержимое пакета Gettext.»

7.8. Bison-3.8.2

Пакет Bison содержит генератор синтаксического анализа.	
Приблизительное время сборки:	0.2 SBU
Требуемое дисковое пространство:	57 MB

7.8.1. Установка пакета Bison

Подготовьте Bison к компиляции:

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

Значение параметров настройки:

- **–docdir=/usr/share/doc/bison-3.8.2**

Этот параметр указывает системе сборки установить документацию к bison в каталог с версией пакета.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Разделе 8.33.2. «Содержимое пакета Bison.»

7.9. Perl-5.38.2

Пакет Perl содержит практический язык для извлечения данных и составления отчётов (Practical Extraction and Report Language).	
Приблизительное время сборки:	0.6 SBU
Требуемое дисковое пространство:	280 MB

7.9.1. Установка пакета Perl

Подготовьте Perl к компиляции:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Duseshrplib                                 \
             -Dprivlib=/usr/lib/perl5/5.38/core_perl      \
             -Darchlib=/usr/lib/perl5/5.38/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.38/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.38/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.38/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.38/vendor_perl
```

Значение параметров настройки:

Значение новых опций Configure:

- **-des**

Это комбинация из трех параметров: -d использует значения по умолчанию для всех элементов; -e обеспечивает выполнение всех задач; -s отключает несущественные выходные данные.

- **-Dvendorprefix=/usr**

Параметр гарантирует, что perl знает, как указать пакетам, где они должны устанавливать свои модули Perl.

- **-Duseshrplib**

Собрать библиотеку libperl, необходимую некоторым модулям Perl, как общую библиотеку вместо статической.

- **-Dprivlib,-Darchlib,-Dsitelib,...**

Эти настройки определяют, где Perl ищет установленные модули. Редакторы LFS решили поместить их в структуру каталогов, основанную на MAJOR.MINOR версии Perl (5.38), что позволяет обновлять Perl до более новых уровней исправлений (уровень исправления - это последняя разделенная точками часть в строке полной версии, например 5.38.2) без необходимости переустанавливать все модули.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Разделе 8.42.2. «Содержимое пакета Perl.»

7.10. Python-3.12.2

Пакет Python 3 содержит среду разработчика Python. Его можно использовать для объектно-ориентированного программирования, написания скриптов, прототипирования больших программ и разработка целых приложений. Python — это интерпретируемый язык программирования.	
Приблизительное время сборки:	0.5 SBU
Требуемое дисковое пространство:	598 MB

7.10.1. Установка пакета Python

Примечание
Существует два пакета, имена которых начинаются с префикса «python». Сейчас необходимо распаковать файл Python-3.12.2.tar.xz (обратите внимание на заглавную первую букву).

Подготовка Python к компиляции:

```
./configure --prefix=/usr   \
            --enable-shared \
            --without-ensurepip
```

Значение параметров настройки:

- **–enable-shared**

Этот параметр отключает установку статичных библиотек.

- **–without-ensurepip**

Этот параметр отключает установщик пакетов Python, который на данном этапе не нужен.

Скомпилируйте пакет:

```
make
```

Примечание
Некоторые модули Python 3 не могут быть собраны сейчас, потому что зависимости еще не установлены. Для модуля **ssl** выводится сообщение **Python требует OpenSSL 1.1.1 или новее**. Сообщение следует проигнорировать. Просто убедитесь, что команда make верхнего уровня не завершилась ошибкой. Дополнительные модули сейчас не нужны, и они будут собраны в Главе 8.

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в Разделе 8.51.2. «Содержимое пакета Python 3.»

7.11. Texinfo-7.1

7.12. Util-linux-2.39.3

7.13. Очистка и сохранение временной системы