Часть II. Подготовка к сборке

2. Подготовка хост-системы

- Пред. chapter01

Содержание

2.1. Введение

В этой главе проверяются и при необходимости устанавливаются основные инструменты, необходимые для построения LFS. Затем подготавливается раздел, в котором будет размещаться система LFS. Мы создадим сам раздел, создадим на нем файловую систему и смонтируем его.

2.2. Требования к хост-системе

2.2.1. Аппаратное обеспечение

Редакторы LFS рекомендуют, чтобы процессор имел не менее четырех ядер и не менее 8 ГБ памяти. Старые системы, не отвечающие этим требованиям, будут по-прежнему работать, но время сборки пакетов будет значительно больше, чем указано в документации.

2.2.2. Программное обеспечение

Ваша хост-система должна иметь следующее программное обеспечение с указанными минимальными версиями. Это не должно быть проблемой для большинства современных дистрибутивов Linux. Также обратите внимание на то, что многие дистрибутивы помещают заголовочные файлы в отдельные пакеты, как правило в формате <package-name>-devel или <package-name>-dev. Обязательно установите эти пакеты, если ваш дистрибутив их предоставляет.

Более ранние версии перечисленных ниже пакетов могут работать, но это не проверялось.

- Bash-3.2 (/bin/sh должен быть символической или жесткой ссылкой на bash)

- Binutils-2.13.1 (Версия выше 2.42 не рекомендуется, так как она не тестировалась)
- Bison-2.7 (/usr/bin/yacc должен быть ссылкой на bison или небольшой скрипт, запускающий bison)
- Coreutils-8.1
- Diffutils-2.8.1
- Findutils-4.2.31
- Gawk-4.0.1 (/usr/bin/awk должен быть ссылкой на gawk)
- GCC-5.2, включая компилятор C++, g++ (версии выше 13.2.0 не рекомендуются, поскольку они не тестировались). Также должны присутствовать стандартные библиотеки C и C++ (с заголовочными файлами), чтобы компилятор C++ мог осуществлять сборку программ.
- Grep-2.5.1a
- Gzip-1.3.12
- Linux Kernel-4.19

Причиной, по которой указаны минимальные требования к версии ядра, является то, что мы указываем эту версию при сборке glibc в Глава 5 и Глава 8. Так как более старые ядра не поддерживаются, скомпилированный пакет glibc немного меньше и быстрее. По состоянию на февраль 2024 г. 4.19 является самой старой версией ядра, поддерживаемой разработчиками ядра. Некоторые версии ядра, более старые, чем 4.19, могут по-прежнему поддерживаться сторонними командами, но они не считаются официальными выпусками ядра; подробности читайте на странице https://kernel.org/category/releases.html

Если версия ядра хоста более ранняя, чем 4.19, вам необходимо обновить ядро на более современную версию. Есть два способа сделать это. Во-первых, посмотрите, предоставляет ли ваш дистрибутив Linux пакет ядра 4.19 или более позднюю версию. Если это так, установите его. Если ваш дистрибутив не предлагает приемлемый пакет ядра или вы предпочитаете не устанавливать его, вы можете скомпилировать ядро самостоятельно. Инструкции по компиляции ядра и настройке загрузчика (при условии, что хост использует GRUB) находятся в Глава 10.

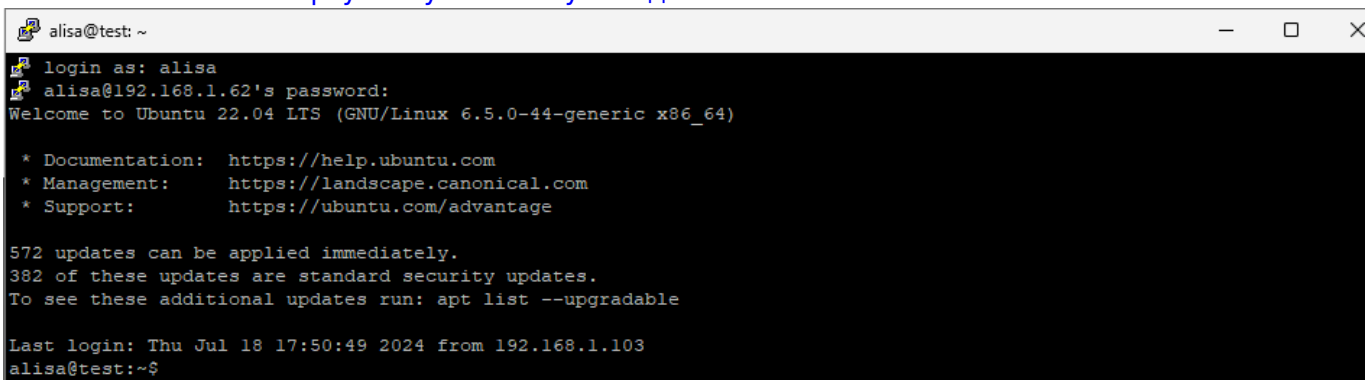
Для сборки LFS необходимо, чтобы ядро хоста поддерживало псевдотерминал UNIX 98 (PTY). Обычно он включен на всех настольных или серверных дистрибутивах, поставляющих Linux 4.19 или более новое ядро. Если на хосте вы используете самостоятельно собранное ядро, убедитесь, что для параметра CONFIG_UNIX98_PTYS установлено значение y в конфигурационном файле ядра.

- M4-1.4.10
- Make-4.0
- Patch-2.5.4
- Perl-5.8.8
- Python-3.4
- Sed-4.1.5
- Tar-1.22
- Texinfo-5.0
- Xz-5.0.0

Важно Обратите внимание, что упомянутые выше символические ссылки необходимы для создания системы LFS с использованием инструкций,

содержащихся в этой книге. Симлинки, указывающие на другое программное обеспечение (например, dash, mawk и т. д.), могут работать, но не тестируются и не поддерживаются командой разработчиков LFS, и могут потребовать либо отклонения от инструкций, либо дополнительных исправлений для некоторых пакетов.

Устыновим linux на вертуальную машину и подключимся по ssh



В моем случае установлена Ubuntu 22.04 LTS desktop amd64 (GNU/Linux 6.5.0-41-generic x86_64)

ubuntu-22.04-desktop-amd64.iso

Для начала получим привилегии суперпользователя

```
sudo su -
```

«См.Подробнее **sudo su -** »

Команда **su** позволяет изменить текущего пользователя терминала на другого. Команда **sudo** выполняет команду от имени root пользователя. Теперь разберем сам вопрос.

sudo su - выполняет команду su от имени суперпользователя. Таким образом, сначала используется sudo чтобы получить права суперпользователя, а затем пользователь меняется на root с помощью su. При этом вы останетесь в той же директории потому, что выполняется ваш .bashrc. А также .profile пользователя root поэтому вы окажетесь в окружении root.

Чтобы узнать, есть ли в вашей хост-системе все необходимые пакеты и возможность компилировать программы, создадим и выполним следующий скрипт:

«Подробности»

cat > version-check.sh « «EOF»

- **cat > « «EOF«** команда создает скрипт **version-check.sh**

#!/bin/bash # A script to list version numbers of critical development …. …. else echo «OK: nproc reports $(nproc) logical cores are available» fi EOF

- Непосредственно код скрипта **version-check.sh**

bash version-check.sh

- Команда выполняет скрипт **version-check.sh**

```
cat > version-check.sh << "EOF"
#!/bin/bash
# A script to list version numbers of critical development tools

# If you have tools installed in other directories, adjust PATH here AND
# in ~lfs/.bashrc (section 4.4) as well.

LC_ALL=C
PATH=/usr/bin:/bin

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep does not work"
sed '' /dev/null || bail "sed does not work"
sort   /dev/null || bail "sort does not work"

ver_check()
{
   if ! type -p $2 &>/dev/null
   then
     echo "ERROR: Cannot find $2 ($1)"; return 1;
   fi
   v=$($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.]+[a-z]*' | head -n1)
   if printf '%s\n' $3 $v | sort --version-sort --check &>/dev/null
   then
     printf "OK:    %-9s %-6s >= $3\n" "$1" "$v"; return 0;
   else
     printf "ERROR: %-9s is TOO OLD ($3 or later required)\n" "$1";
     return 1;
   fi
}

ver_kernel()
{
   kver=$(uname -r | grep -E -o '^[0-9\.]+')
   if printf '%s\n' $1 $kver | sort --version-sort --check &>/dev/null
   then
     printf "OK:    Linux Kernel $kver >= $1\n"; return 0;
   else
     printf "ERROR: Linux Kernel ($kver) is TOO OLD ($1 or later
required)\n" "$kver";
     return 1;
   fi
}

# Coreutils first because --version-sort needs Coreutils >= 7.0
```

```
ver_check Coreutils      sort     8.1 || bail "Coreutils too old, stop"
ver_check Bash           bash     3.2
ver_check Binutils       ld       2.13.1
ver_check Bison          bison    2.7
ver_check Diffutils      diff     2.8.1
ver_check Findutils      find     4.2.31
ver_check Gawk           gawk     4.0.1
ver_check GCC            gcc      5.2
ver_check "GCC (C++)"    g++      5.2
ver_check Grep           grep     2.5.1a
ver_check Gzip           gzip     1.3.12
ver_check M4             m4       1.4.10
ver_check Make           make     4.0
ver_check Patch          patch    2.5.4
ver_check Perl           perl     5.8.8
ver_check Python         python3  3.4
ver_check Sed            sed      4.1.5
ver_check Tar            tar      1.22
ver_check Texinfo        texi2any 5.0
ver_check Xz             xz       5.0.0
ver_kernel 4.19

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK:    Linux Kernel supports UNIX 98 PTY";
else echo "ERROR: Linux Kernel does NOT support UNIX 98 PTY"; fi

alias_check() {
   if $1 --version 2>&1 | grep -qi $2
   then printf "OK:    %-4s is $2\n" "$1";
   else printf "ERROR: %-4s is NOT $2\n" "$1"; fi
}
echo "Aliases:"
alias_check awk GNU
alias_check yacc Bison
alias_check sh Bash

echo "Compiler check:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK:    g++ works";
else echo "ERROR: g++ does NOT work"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
   echo "ERROR: nproc is not available or it produces empty output"
else
   echo "OK: nproc reports $(nproc) logical cores are available"
fi
EOF

bash version-check.sh
```

```
OK:    Coreutils 8.32   >= 8.1
OK:    Bash      5.1.16 >= 3.2
ERROR: Cannot find ld (Binutils)
ERROR: Cannot find bison (Bison)
OK:    Diffutils 3.8    >= 2.8.1
OK:    Findutils 4.8.0  >= 4.2.31
ERROR: Cannot find gawk (Gawk)
ERROR: Cannot find gcc (GCC)
ERROR: Cannot find g++ (GCC (C++))
OK:    Grep      3.7    >= 2.5.1a
OK:    Gzip      1.10   >= 1.3.12
ERROR: Cannot find m4 (M4)
ERROR: Cannot find make (Make)
OK:    Patch     2.7.6  >= 2.5.4
OK:    Perl      5.34.0 >= 5.8.8
OK:    Python    3.10.4 >= 3.4
OK:    Sed       4.8    >= 4.1.5
OK:    Tar       1.34   >= 1.22
ERROR: Cannot find texi2any (Texinfo)
OK:    Xz        5.2.5  >= 5.0.0
OK:    Linux Kernel 6.5.0 >= 4.19
OK:    Linux Kernel supports UNIX 98 PTY
Aliases:
ERROR: awk  is NOT GNU
ERROR: yacc is NOT Bison
ERROR: sh   is NOT Bash
Compiler check:
version-check.sh: line 81: g++: command not found
ERROR: g++ does NOT work
OK: nproc reports 8 logical cores are available
```

Система выводит нам ошибки

```
ОШИБКА: Невозможно найти ld (Binutils)
ОШИБКА: Невозможно найти бизона (Бизон).
OK: Diffutils 3.8 >= 2.8.1
OK: Findutils 4.8.0 >= 4.2.31
ОШИБКА: Невозможно найти Gawk (Gawk)
ОШИБКА: Невозможно найти gcc (GCC)
ОШИБКА: Невозможно найти g++ (GCC (C++))
OK: Grep 3.7 >= 2.5.1a
OK: Gzip 1.10 >= 1.3.12
ОШИБКА: Невозможно найти m4 (M4)
ОШИБКА: Невозможно найти марку (Make)
OK: Патч 2.7.6 >= 2.5.4
OK: Perl 5.34.0 >= 5.8.8
OK: Python 3.10.4 >= 3.4
OK: Сед 4.8 >= 4.1.5
OK: Tar 1.34 >= 1.22
ОШИБКА: Невозможно найти texi2any (Texinfo)
OK: Xz 5.2.5 >= 5.0.0
OK: ядро Linux 6.5.0 >= 4.19
OK: ядро Linux поддерживает UNIX 98 PTY.
Псевдонимы:
ОШИБКА: awk НЕ GNU
ОШИБКА: yacc НЕ Bison
ОШИБКА: sh НЕ Bash
Проверка компилятора:
version-check.sh: строка 81: g++: команда не найдена
ОШИБКА: g++ НЕ работает
OK: nproc сообщает, что доступно 8 логических ядер.
```

Обновите пакеты с помощью приведенной ниже команды, затем перезагрузитесь:

```
apt update; apt upgrade -y; reboot
```

```
root@test:~# apt update; apt upgrade -y; reboot
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
```

При перезагрузке система вас отключит от сессии. Подсключитесь снова после перезагрузки по ssh. Затем удалите все ненужные пакеты, используя:

```
apt --purge autoremove
```

```
root@test:~# apt --purge autoremove
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be REMOVED:
  libflashrom1* libftdi1-2* libllvm13*
0 upgraded, 0 newly installed, 3 to remove and 3 not upgraded.
After this operation, 101 MB disk space will be freed.
Do you want to continue? [Y/n]
```

Соглашаемся продолжить, нажитием клавиши Y и подтверждаем клавишей Enter

```
Do you want to continue? [Y/n] y
(Reading database ... 200897 files and directories currently installed.)
Removing libflashrom1:amd64 (1.2-5build1) ...
Removing libftdi1-2:amd64 (1.5-5build3) ...
Removing libllvm13:amd64 (1:13.0.1-2ubuntu2.2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
root@test:~#
```

Устанавливаем Binutils (binutils-source является колекцией средств разработки программ, содержащих компоновщик, ассемблер и другие средства для работы с объектными файлами) с помощью команды

```
apt-get -y install binutils
```

В процесе инсталяции будут проверенны и установленны зависимости зависимости установки: Bash, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl, Sed и Texinfo

```
Setting up libbinutils:amd64 (2.38-4ubuntu2.6) ...
Setting up libctf0:amd64 (2.38-4ubuntu2.6) ...
Setting up binutils-x86-64-linux-gnu (2.38-4ubuntu2.6) ...
Setting up binutils (2.38-4ubuntu2.6) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Processing triggers for man-db (2.10.2-1) ...
root@test:~#
```

Снова воспользуемся скриптом для проверки нашей хост-системы

```
bash version-check.sh
```

```
root@test:~# bash version-check.sh
```

Проверяем результат

```
OK:    Bash      5.1.16 >= 3.2
OK:    Binutils  2.38   >= 2.13.1
ERROR: Cannot find bison (Bison)
OK:    Diffutils 3.8    >= 2.8.1
OK:    Findutils 4.8.0  >= 4.2.31
ERROR: Cannot find gawk (Gawk)
ERROR: Cannot find gcc (GCC)
ERROR: Cannot find g++ (GCC (C++))
OK:    Grep      3.7    >= 2.5.1a
OK:    Gzip      1.10   >= 1.3.12
ERROR: Cannot find m4 (M4)
ERROR: Cannot find make (Make)
OK:    Patch     2.7.6  >= 2.5.4
OK:    Perl      5.34.0 >= 5.8.8
OK:    Python    3.10.12 >= 3.4
OK:    Sed       4.8    >= 4.1.5
OK:    Tar       1.34   >= 1.22
ERROR: Cannot find texi2any (Texinfo)
OK:    Xz        5.2.5  >= 5.0.0
OK:    Linux Kernel 6.5.0 >= 4.19
OK:    Linux Kernel supports UNIX 98 PTY
Aliases:
ERROR: awk  is NOT GNU
ERROR: yacc is NOT Bison
ERROR: sh   is NOT Bash
Compiler check:
version-check.sh: line 81: g++: command not found
ERROR: g++ does NOT work
OK: nproc reports 8 logical cores are available
root@test:~#
```

Видим установленный Binutils версии 2.38, а так же установившиеся пакеты: Make, Texinfo. Если это не так, используем следующую команду для установки

```
apt-get -y install make texinfo
```

```
root@test:~# apt-get -y install make texinfo
```

Для установки остальных пакетов воспользуемся командой

```
apt-get -y install bison gawk gcc g++ m4
```

```
root@test:~# apt-get -y install bison gawk gcc g++ m4
```

Процесс довольно быстрый с примерным выводом установленных пакетов приведен ниже

```
Setting up libitm1:amd64 (12.3.0-1ubuntu1~22.04) ...
Setting up libc-devtools (2.35-0ubuntu3.8) ...
Setting up libtsan0:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up libgcc-11-dev:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up gcc-11 (11.4.0-1ubuntu1~22.04) ...
Setting up libc6-dev:amd64 (2.35-0ubuntu3.8) ...
Setting up gcc (4:11.2.0-1ubuntu1) ...
Setting up libstdc++-11-dev:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up g++-11 (11.4.0-1ubuntu1~22.04) ...
Setting up g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
root@test:~#
```

Обновим пакеты с помощью приведенной ниже команды:

```
apt update; apt upgrade -y
```

Снова воспользуемся скриптом для проверки нашей хост-системы

```
bash version-check.sh
```

```
root@test:~# bash version-check.sh
```

Проверяем результат

```
OK:    Coreutils 8.32   >= 8.1
OK:    Bash      5.1.16 >= 3.2
OK:    Binutils  2.38   >= 2.13.1
OK:    Bison     3.8.2  >= 2.7
OK:    Diffutils 3.8    >= 2.8.1
OK:    Findutils 4.8.0  >= 4.2.31
OK:    Gawk      5.1.0  >= 4.0.1
OK:    GCC       11.4.0 >= 5.2
OK:    GCC (C++) 11.4.0 >= 5.2
OK:    Grep      3.7    >= 2.5.1a
OK:    Gzip      1.10   >= 1.3.12
OK:    M4        1.4.18 >= 1.4.10
OK:    Make      4.3    >= 4.0
OK:    Patch     2.7.6  >= 2.5.4
OK:    Perl      5.34.0 >= 5.8.8
OK:    Python    3.10.12 >= 3.4
OK:    Sed       4.8    >= 4.1.5
OK:    Tar       1.34   >= 1.22
OK:    Texinfo   6.8    >= 5.0
OK:    Xz        5.2.5  >= 5.0.0
OK:    Linux Kernel 6.5.0 >= 4.19
OK:    Linux Kernel supports UNIX 98 PTY
Aliases:
OK:    awk  is GNU
OK:    yacc is Bison
ERROR: sh   is NOT Bash
Compiler check:
OK:    g++ works
OK: nproc reports 8 logical cores are available
root@test:~#
```

Осталось устранить ошибку «ОШИБКА: sh НЕ Bash»

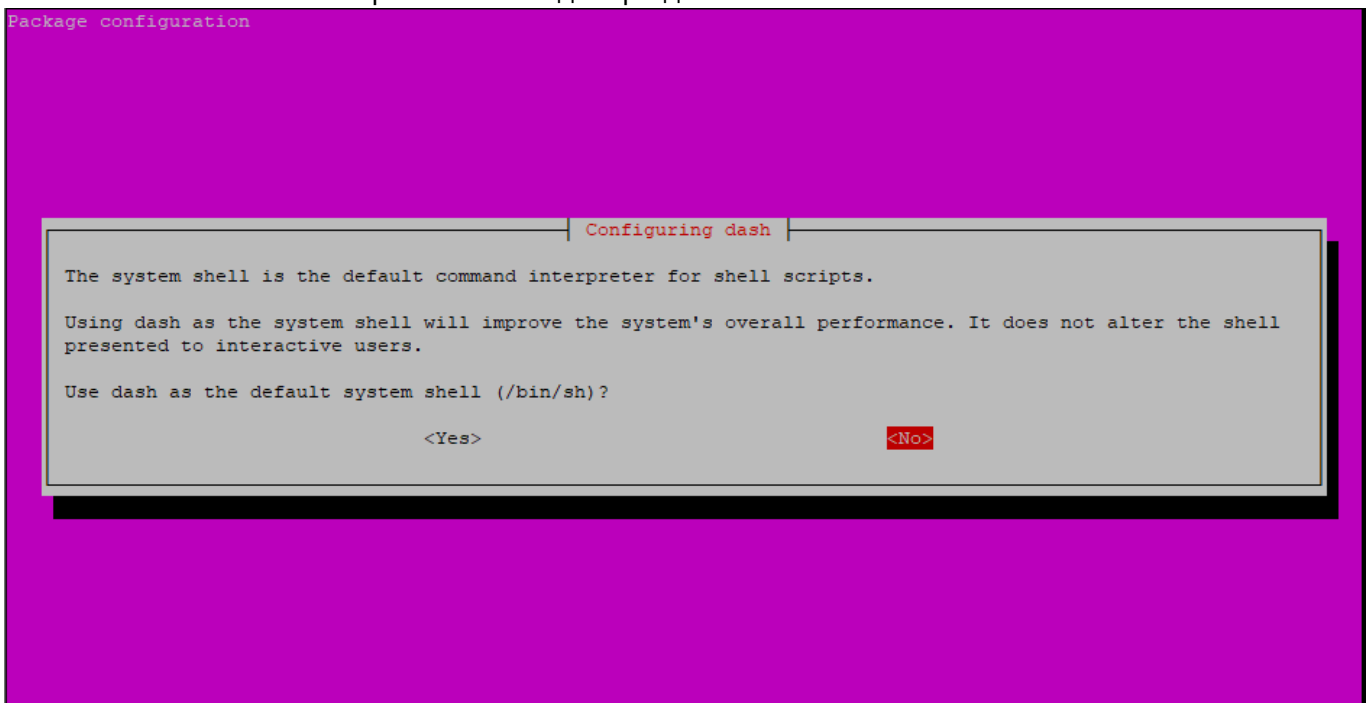
```
ERROR: sh   is NOT Bash
```

В Ubuntu /bin/sh это dash, который разработан для быстрой работы, использования небольшого объема памяти и не поддерживает намного больше, чем минимум, ожидаемый от /bin/sh

```
dpkg-reconfigure dash
```

```
root@test:~# dpkg-reconfigure dash
```

В появившемся окне выбираем NO и подтверждаем клавишей Enter



Видим ответ системы об удалении и добавлении перенаправлений с dash на bash

```
root@test:~# dpkg-reconfigure dash
Removing 'diversion of /bin/sh to /bin/sh.distrib by dash'
Adding 'diversion of /bin/sh to /bin/sh.distrib by bash'
Removing 'diversion of /usr/share/man/man1/sh.1.gz to /usr/share/man/man1/sh.distrib.1.gz by dash'
Adding 'diversion of /usr/share/man/man1/sh.1.gz to /usr/share/man/man1/sh.distrib.1.gz by bash'
root@test:~#
```

В последний раз запускаем скрипт проверки готовности нашей хост-системы.

```
bash version-check.sh
```

```
root@test:~# bash version-check.sh
```

Проверяем результат

```
root@test:~# bash version-check.sh
OK:    Coreutils 8.32   >= 8.1
OK:    Bash      5.1.16 >= 3.2
OK:    Binutils  2.38   >= 2.13.1
OK:    Bison     3.8.2  >= 2.7
OK:    Diffutils 3.8    >= 2.8.1
OK:    Findutils 4.8.0  >= 4.2.31
OK:    Gawk      5.1.0  >= 4.0.1
OK:    GCC       11.4.0 >= 5.2
OK:    GCC (C++) 11.4.0 >= 5.2
OK:    Grep      3.7    >= 2.5.1a
OK:    Gzip      1.10   >= 1.3.12
OK:    M4        1.4.18 >= 1.4.10
OK:    Make      4.3    >= 4.0
OK:    Patch     2.7.6  >= 2.5.4
OK:    Perl      5.34.0 >= 5.8.8
OK:    Python    3.10.12 >= 3.4
OK:    Sed       4.8    >= 4.1.5
OK:    Tar       1.34   >= 1.22
OK:    Texinfo   6.8    >= 5.0
OK:    Xz        5.2.5  >= 5.0.0
OK:    Linux Kernel 6.5.0 >= 4.19
OK:    Linux Kernel supports UNIX 98 PTY
Aliases:
OK:    awk  is GNU
OK:    yacc is Bison
OK:    sh   is Bash
Compiler check:
OK:    g++ works
OK: nproc reports 8 logical cores are available
root@test:~#
```

2.3. Этапы сборки системы LFS

LFS разработан для сборки за один сеанс. То есть инструкция предполагает, что система не будет выключаться в процессе. Это не означает, что система должна быть собрана за один присест. Для возобновления сборки в точке предыдущей остановки (после перезагрузки/выключения), необходимо выполнить некоторые процедуры повторно.

2.3.1. Главы 1–4

Эти главы выполняются на хост-системе. После перезагрузки обратите внимание на следующее:

При выполнении операций, от имени пользователя root после Раздела 2.4, ДЛЯ ПОЛЬЗОВАТЕЛЯ root должна быть установлена переменная окружения LFS.

2.3.2. Главы 5–6

Раздел /mnt/lfs должен быть смонтирован.

Эти две главы должны быть выполнены из-под пользователя lfs. Перед выполнением любой задачи в этих главах необходимо выполнить команду su - lfs. В противном случае вы рискуете установить пакеты на хост и сделать его непригодным для использования.

Выполнение процедур из Общие инструкции по компиляции имеет решающее значение. Если есть какие-либо сомнения по поводу установки пакета, убедитесь, что все ранее распакованные tar-архивы удалены, затем повторно извлеките файлы и выполните все инструкции, приведенные в этом разделе.

2.3.3. Главы 7–10

Раздел /mnt/lfs должен быть смонтирован.

Некоторые операции, такие как «Смена владельца» или «Вход в среду Chroot», должны быть выполнены от имени пользователя root с переменной окружения $LFS, установленной для пользователя root.

При входе в chroot переменная среды LFS должна быть установлена для пользователя root. Переменная LFS не используется после входа в среду chroot.

Виртуальные файловые системы должны быть смонтированы. Это можно сделать до или после входа в chroot, переключившись на виртуальный терминал хоста и от имени пользователя root выполнив команды, описанные в Раздел 7.3.1, «Монтирование и заполнение /dev» и Раздел 7.3.2, «Монтирование виртуальных файловых систем ядра».

2.4. Создание нового раздела

Как и большинство других операционных систем, LFS обычно устанавливается на выделенный раздел. Рекомендуемый подход к построению системы LFS состоит в том, чтобы использовать доступный пустой раздел или, если у вас достаточно неразмеченного пространства, использовать его

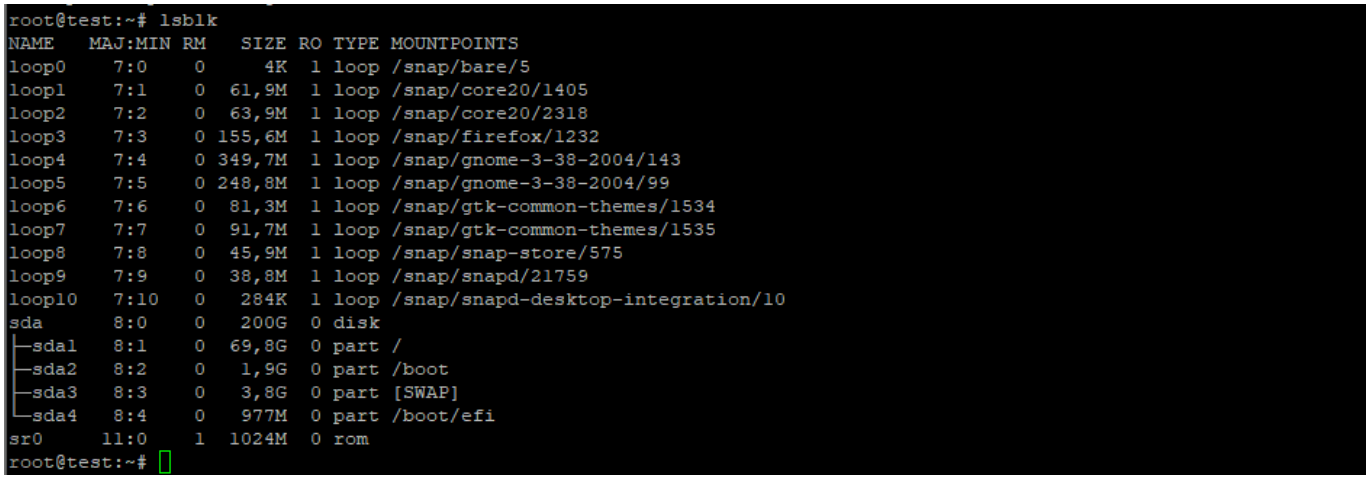
Минимальная система требует раздел размером около 10 гигабайт (ГБ). Этого достаточно для хранения всех архивов с исходным кодом и компиляции пакетов. Однако, если система LFS предназначена для использования в качестве основной системы Linux, вероятно, будет установлено дополнительное программное обеспечение, для которого потребуется дополнительное пространство. Раздел размером 30 ГБ является разумным размером для расширения. Сама система LFS не займет столько места. Большая часть этого требования заключается в предоставлении достаточного временного хранилища, а также в добавлении дополнительных возможностей после сборки LFS. Кроме того, для компиляции пакетов может потребоваться много места на диске, которое будет освобождено после установки пакета.

При создании виртуальной машины в мастере создания виртуальных машин, мы создали виртуальный диск размером **200Gb** а при установке Ubuntu на виртуальную машину Hyper-V

мы выделили размер диска sda1 в **75Gb** (для установки программ из книги BLFS в будущем), при этом размер диска sda в **132,6Gb** оставили неразмеченным.

На неразмеченном пространстве нашего диска sda выделим раздел для построения нашей будущей системы LFS, для этого выведим список устройств хранения и разделов командой:

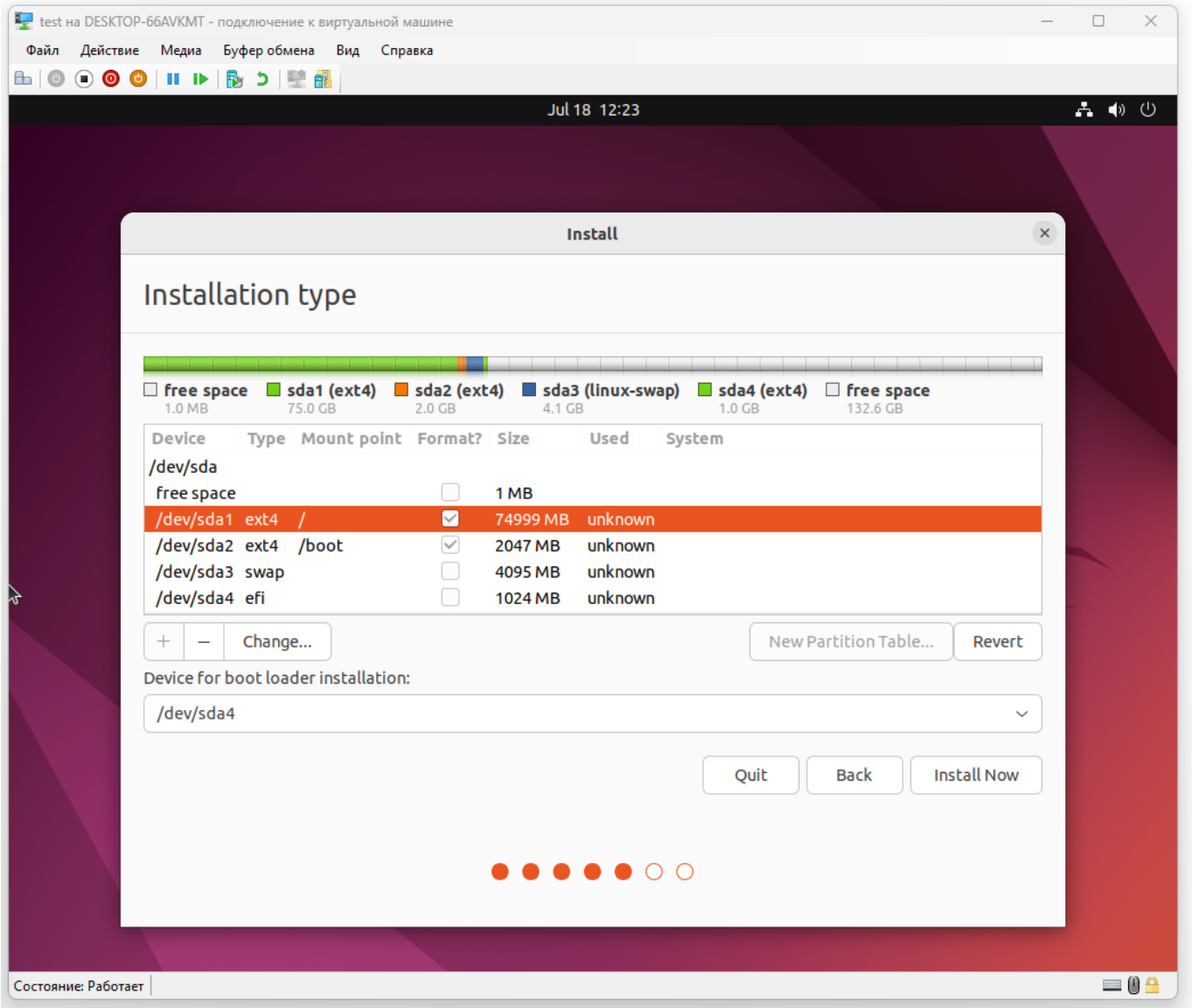
```
lsblk
```



Мы видим общий размер нашего диска **sda** в **200G** и раздела **sda1** в **69,8G**, а так же разделы **/boot, SWAP** и **/boot/efi**, созданные нами при установке Ubuntu.

```
sda      8:0    0   200G  0 disk
├─sda1   8:1    0  69,8G  0 part /
├─sda2   8:2    0   1,9G  0 part /boot
├─sda3   8:3    0   3,8G  0 part [SWAP]
└─sda4   8:4    0   977M  0 part /boot/efi
```

Нами была оставлена неразмеченная область **free space** размером **132.6Gb**

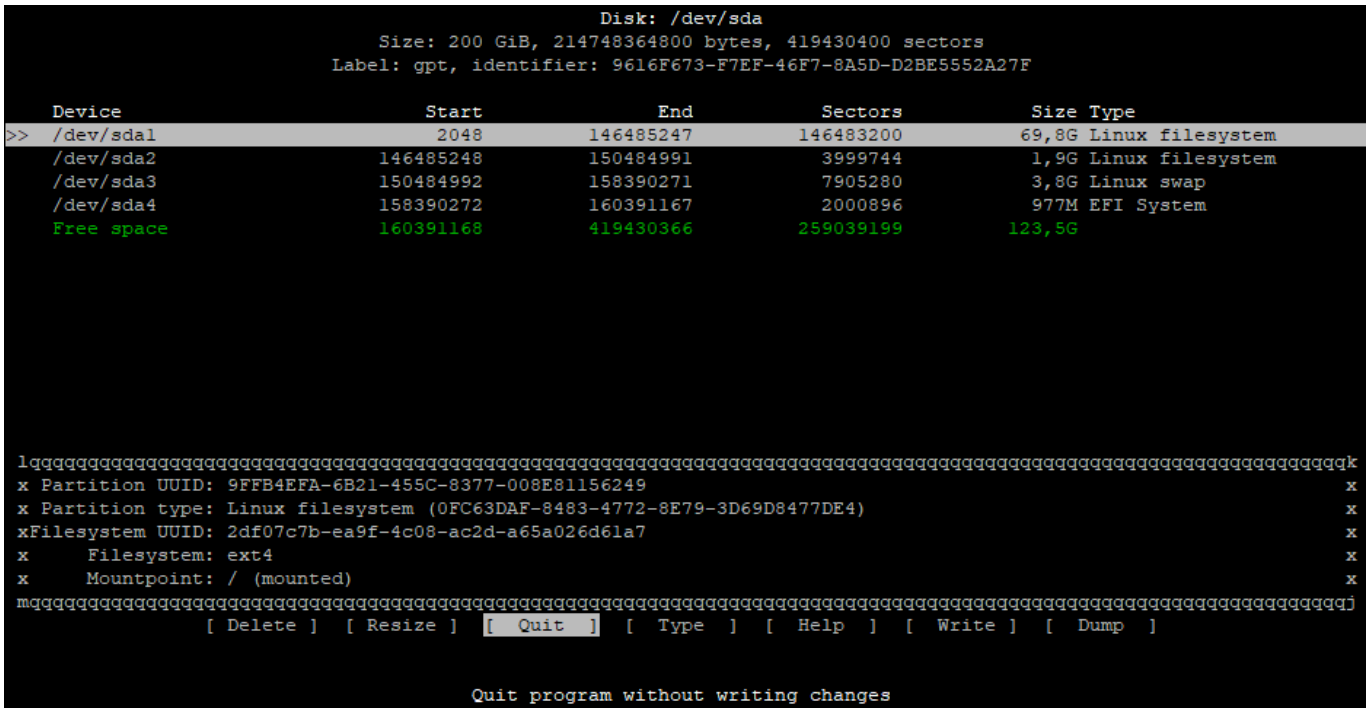


Пришло время создать раздел на устройстве **/dev/sda** в этой области. Воспользуемся утилитой cfdisk

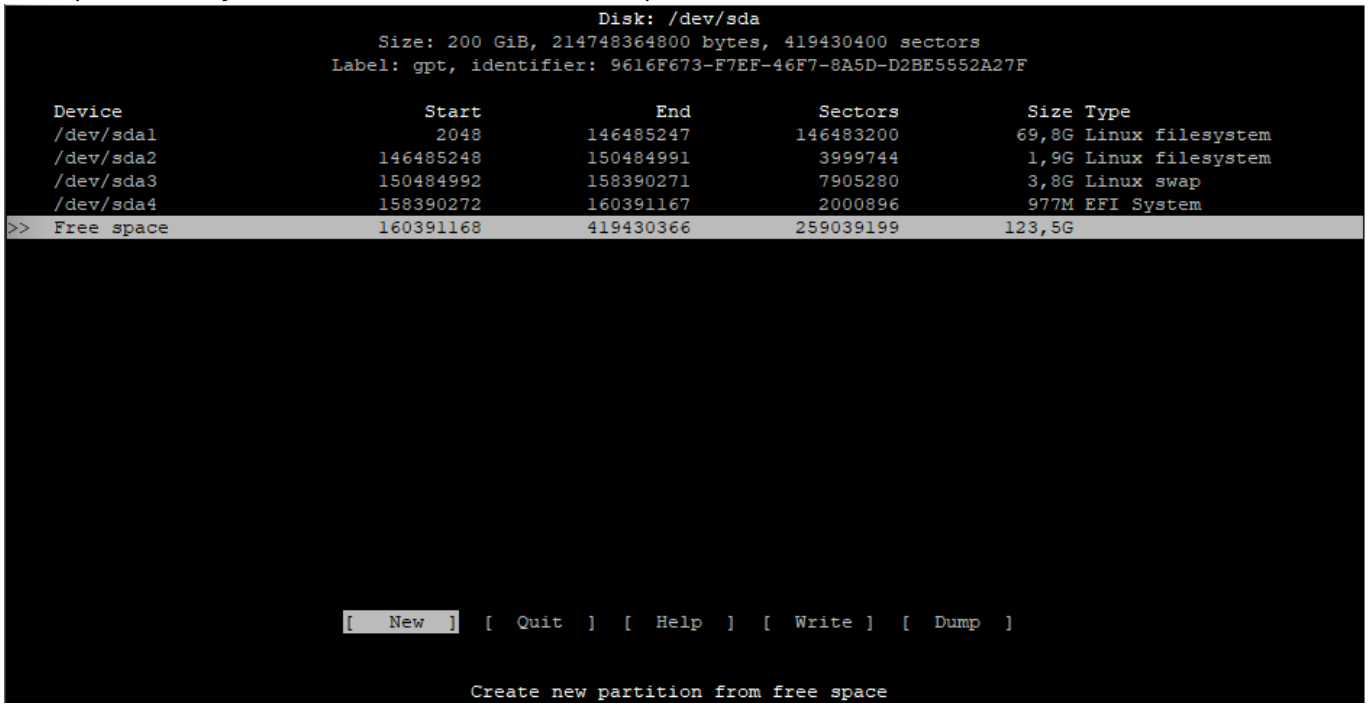
```
cfdisk /dev/sda
```

```
root@test:~# cfdisk /dev/sda
```

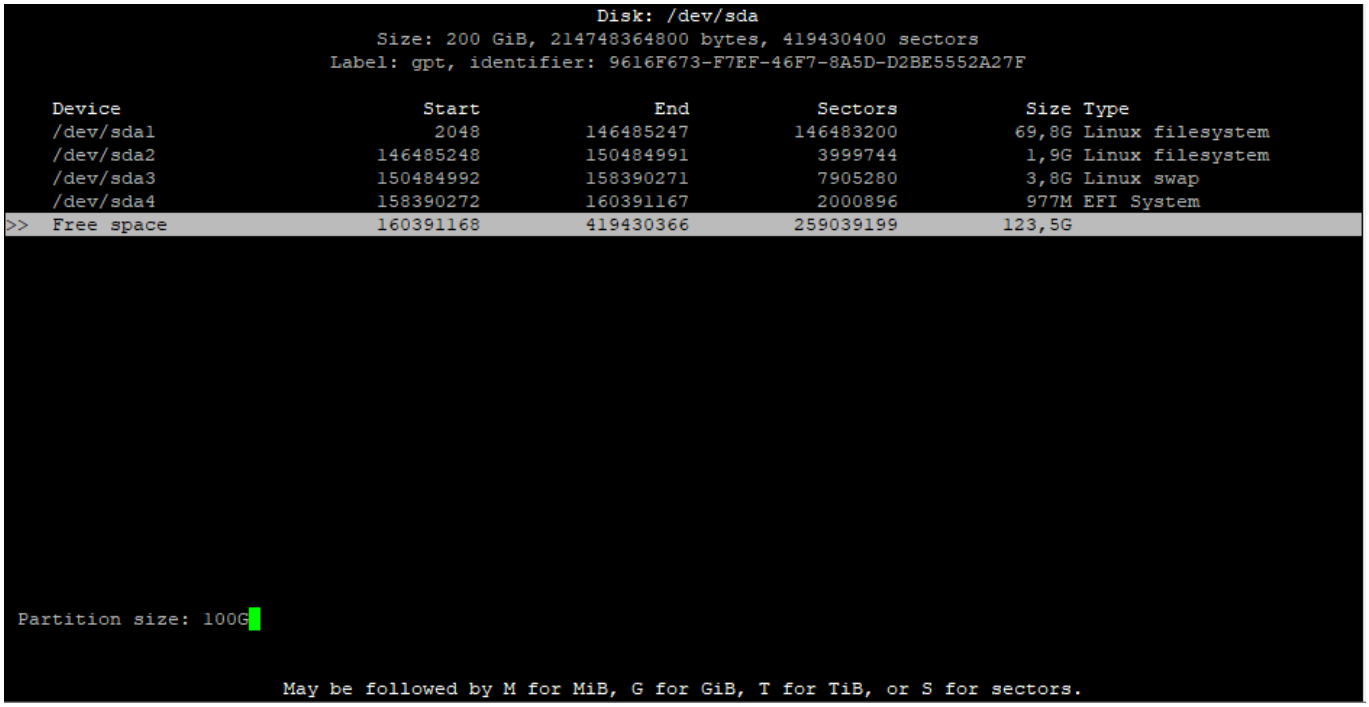
воспользуемся интерактивным интерфейсом программы cfdisk



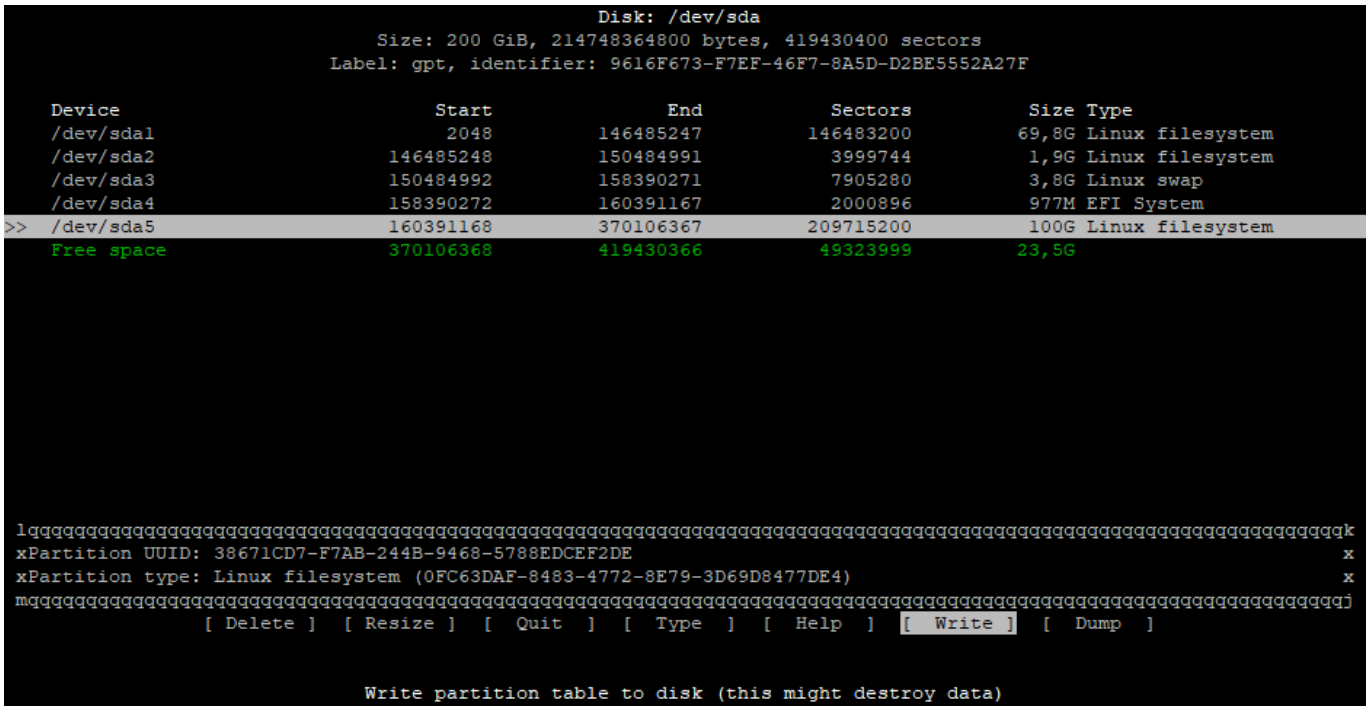
Стрелочкой вниз перейдем на неразмеченную область **free space**, стрелкой лево/право выбирем Кнопку «New» для создания нового раздела



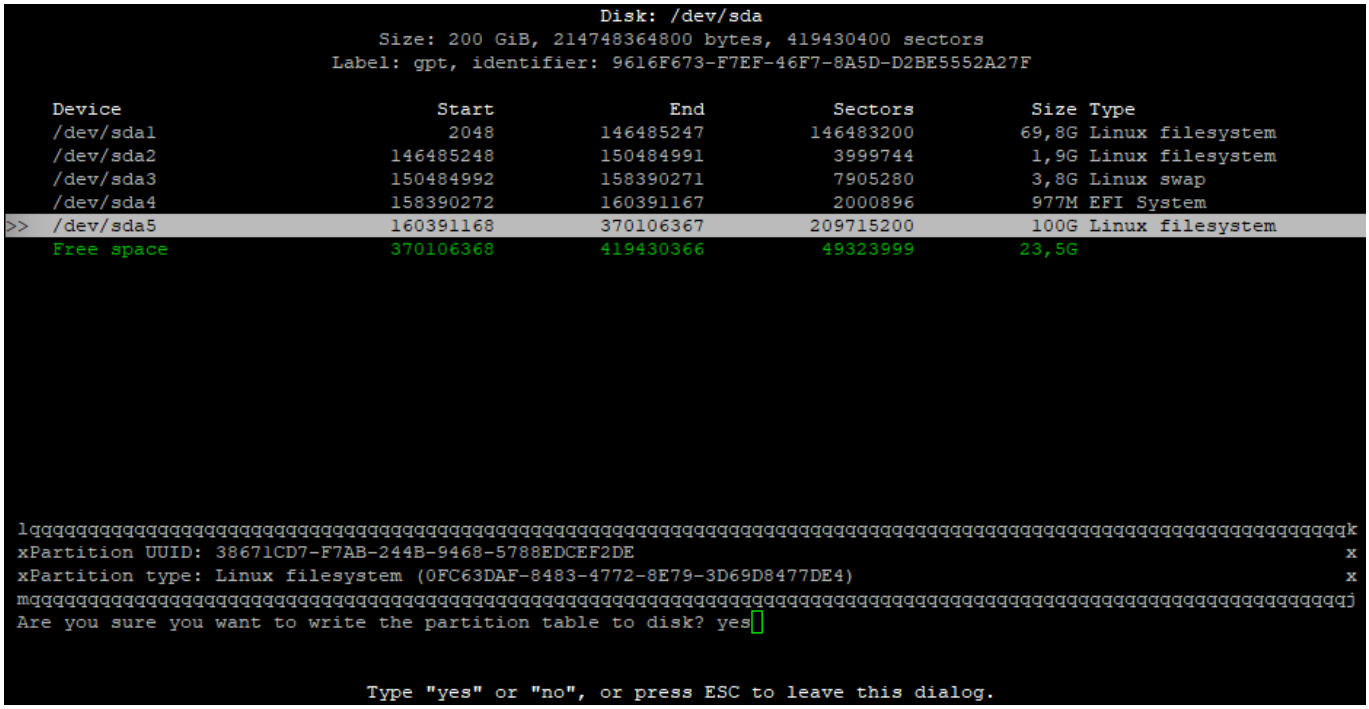
Укажем размер создаваемого раздела, оставив небольшой **free space** про запас. Подтвердим клавишей **Enter**



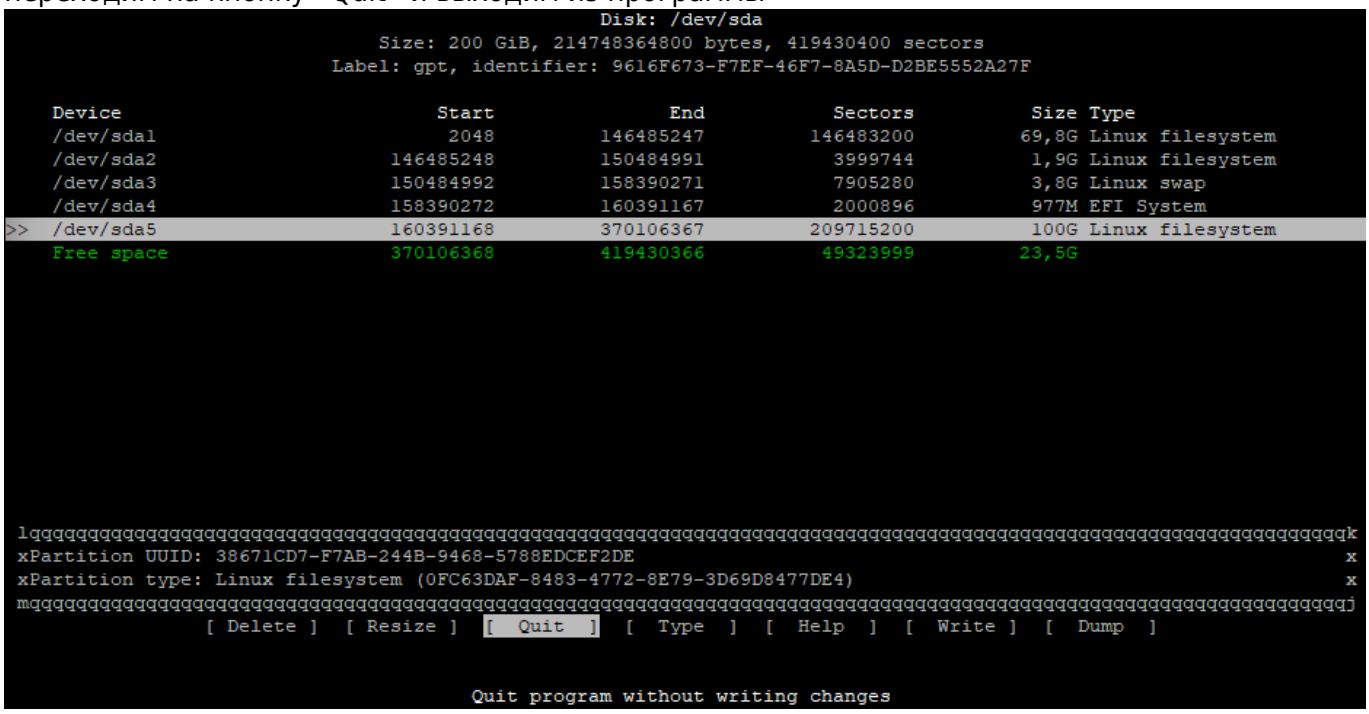
стрелкой вправо выбирем Кнопку «Write» для создания нового раздела. Подтвердим клавишей **Enter**



На вопрос программы «Are you sure you want to write the partition table to disk?» отвечаем «yes»

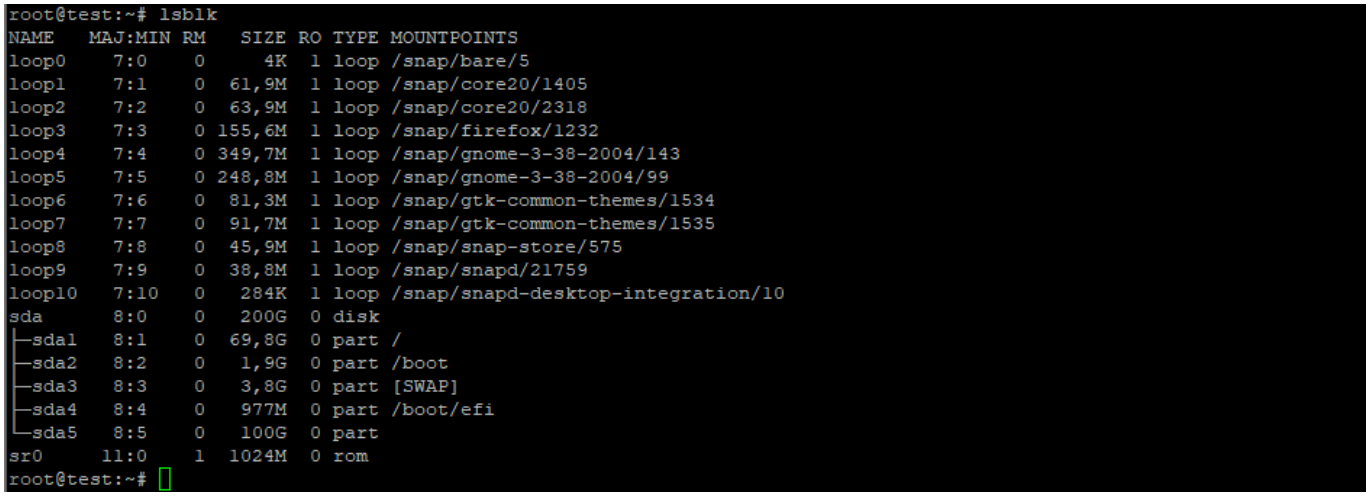


переходим на кнопку «Quit» и выходим из программы



Проконтролируем новую таблицу разделов:

```
lsblk
```



Запомним имена разделов: **sda3** размером 3,8G [**SWAP**] и новый **sda5** размером **100Gb**. В этой книге новый **sda5** будет называться разделом **LFS**. Эти имена понадобятся позже для файла /etc/fstab.

```
├─sda3   8:3    0   3,8G  0 part [SWAP]
├─sda4   8:4    0   977M  0 part /boot/efi
└─sda5   8:5    0   100G  0 part
```

Поскольку для компиляции не всегда достаточно оперативной памяти (ОЗУ), рекомендуется использовать небольшой раздел диска в качестве раздела подкачки. Он используется ядром для хранения редко используемых данных и оставляет больше памяти для активных процессов. Раздел подкачки для системы LFS может совпадать с разделом, используемым хост-системой, и в этом случае нет необходимости создавать еще один.

Запустите программу создания разделов диска, такую как cfdisk или fdisk, с параметром командной строки, указав имя жесткого диска, на котором будет создан новый раздел, например, /dev/sda для основного диска. Создайте раздел Linux и раздел подкачки, если это необходимо. Пожалуйста, обратитесь к справке cfdisk(8) или fdisk(8), если вы еще не знаете, как пользоваться этими программами.

> Примечание Для опытных пользователей возможны и другие схемы разбиения. Система LFS может располагаться на программном RAID-массиве или логическом томе LVM. Однако для некоторых опций требуется initramfs, что является сложной темой. Эти методы разбиения не рекомендуются начинающим пользователям LFS.

2.4.1. Другие вопросы по созданию разделов

Рекомендации по созданию разделов системы часто публикуются в списках рассылки LFS. Это очень субъективная тема. По умолчанию для большинства дистрибутивов используется весь диск, за исключением небольшого раздела подкачки. Это не оптимально для LFS по нескольким причинам. Это снижает гибкость, затрудняет совместное использование данных между несколькими дистрибутивами или сборками LFS, делает резервное копирование более трудоемким и может тратить дисковое пространство из-за неэффективно распределенной файловой системы.

2.4.1.1. Корневой раздел

Корневой раздел LFS (не путать с каталогом /root) размером в 20 гигабайт является хорошим компромиссом для большинства систем. Он обеспечивает достаточно места для построения LFS и большей части BLFS, но достаточно мал, чтобы можно было легко создать несколько разделов для экспериментов.

2.4.1.2. Раздел подкачки

Большинство дистрибутивов автоматически создают раздел подкачки. Обычно рекомендуемый размер раздела подкачки примерно в два раза превышает объем физической памяти, однако это требуется редко. Если дисковое пространство ограничено, установите размер раздела подкачки в два гигабайта и контролируйте его объемом.

Если вы хотите использовать режим гибернации (suspend-to-disk) Linux, которая записывает содержимое ОЗУ в раздел подкачки перед выключением машины. Установите размер раздела подкачки не меньше объема установленной оперативной памяти.

Использование файла подкачки - это не очень хорошо. Для механических жестких дисков вы можете определить, что система использует раздел подкачки, просто слыша активность диска и наблюдая, как система реагирует на команды. Для SSD-накопителя вы не сможете услышать, что используется раздел подкачки, но сможете оценить, сколько места на разделе подкачки занято, используя команды top или free. По возможности следует избегать использования SSD-накопителя для раздела подкачки. Первой реакцией на активность раздела подкачки должна быть проверка на необоснованное применение какой-либо команды, например, попытка редактирования пятигигабайтного файла. Если использование раздела подкачки становится обычным явлением, лучшее решение — приобретение большего объема оперативной памяти для вашей системы.

2.4.1.3. Раздел GRUB

Если загрузочный диск размечен с помощью таблицы разделов GUID (GPT), необходимо создать небольшой раздел, обычно размером 1 МБ, если он еще не существует. Этот раздел не форматируется, но должен быть доступен для использования GRUB во время установки загрузчика. Обычно он помечен как 'BIOS Boot' при использовании fdisk или имеет код EF02 при использовании gdisk.

[Примечание] Примечание Раздел Grub Bios должен находиться на диске, который BIOS использует для загрузки системы. Это не обязательно тот же диск, на котором расположен корневой раздел LFS. Диски в системе могут использовать разные типы таблиц разделов. Наличие раздела Grub Bios зависит только от типа таблицы разделов на загрузочном диске.

2.4.1.4. Разделы, используемые для удобства

Есть несколько других разделов, которые не являются обязательными, но их следует

учитывать при разработке схемы диска. Следующий список не является исчерпывающим, а представлен в качестве справочного руководства.

- /boot – Настоятельно рекомендуется. Используйте этот раздел для хранения ядер и другой загрузочной информации. Чтобы свести к минимуму возможные проблемы с загрузкой дисков большого размера, сделайте этот раздел первым физическим разделом на первом диске. Размер раздела в 200 мегабайт вполне достаточен.
- /boot/efi – Системный раздел EFI, используемый для загрузки системы с помощью UEFI. Подробнее читайте на странице BLFS.
- /home – Настоятельно рекомендуется. Предоставьте общий доступ к своему домашнему каталогу и пользовательским настройкам нескольким дистрибутивам или сборкам LFS. Размер, как правило, довольно большой и зависит от доступного места на диске.
- /usr – в LFS, /bin, /lib, и /sbin являются символическими ссылками на их аналоги в /usr. Таким образом /usr содержит все двоичные файлы, необходимые для работы системы. Для LFS отдельный раздел /usr не требуется. Если он вам необходим, вы должны сделать раздел достаточно большим, чтобы поместить туда все программы и библиотеки в системе. В этой конфигурации, корневой раздел может быть очень маленьким (возможно, всего один гигабайт), поэтому он подходит для тонкого клиента или бездисковой рабочей станции (где /usr монтируется с удаленного сервера). Однако вы должны знать, что для загрузки системы с отдельного раздела /usr потребуется initramfs (не включенный в LFS).
- /opt – Этот каталог наиболее полезен для BLFS, в него можно установить некоторые большие пакеты, такие как KDE или Texlive, без использования иерархии /usr. Для /opt достаточно размера от 5 до 10 гигабайт.
- /tmp – По умолчанию, systemd монтирует здесь tmpfs. Если вы хотите переопределить это поведение, следуйте инструкции Раздел 9.10.3, «Отключение tmpfs для /tmp» при настройке системы LFS.
- /usr/src – Этот раздел очень удобен для хранения исходников BLFS и совместного использования их в сборках LFS. Его также можно использовать в качестве места для сборки пакетов BLFS. Размера в 30-50 гигабайт вполне достаточно.

Любой отдельный раздел, который вы хотите автоматически монтировать при загрузке, должен быть указан в файле /etc/fstab. Подробности о том, как указать разделы, будут обсуждаться в Раздел 10.2, «Создание файла /etc/fstab».

2.5. Создание файловой системы на разделе

Раздел - это всего лишь диапазон секторов на диске, указанный в таблице разделов. Прежде чем операционная система сможет использовать раздел для хранения каких-либо файлов, он должен быть отформатирован, чтобы содержать файловую систему, обычно состоящую из метки, блоков каталогов, блоков данных и схемы индексации для поиска конкретного файла по запросу. Файловая система также помогает операционной системе отслеживать свободное пространство на разделе, резервировать необходимые секторы при создании нового файла или расширении существующего и повторно использует свободные сегменты данных, полученные в результате удаления файлов. Она также может обеспечивать поддержку избыточности данных и восстановления после ошибок.

LFS может использовать любую файловую систему, распознаваемую ядром Linux, но наиболее распространенными типами являются ext3 и ext4. Выбор правильной файловой системы может быть сложным; это зависит от характеристик файлов и размера раздела. Например:

ext2	подходит для небольших разделов, которые редко обновляются, например /boot.
ext3	это обновленная файловая система ext2, которая включает в себя журнал, помогающий восстановить состояние раздела в случае некорректного завершения работы. Обычно используется в качестве файловой системы общего назначения.
ext4	является последней версией файловых систем семейства ext. Она предоставляет несколько новых возможностей, включая временные метки с точностью до наносекунды, создание и использование очень больших файлов (16 ТБ) и повышение скорости работы.

Другие файловые системы, включая FAT32, NTFS, ReiserFS, JFS и XFS, полезны для конкретных задач. Более подробную информацию об этих файловых системах и многих других можно найти по адресу https://en.wikipedia.org/wiki/Comparison_of_file_systems.

LFS предполагает, что корневая файловая система (/) имеет тип ext4. Чтобы создать файловую систему ext4 на разделе LFS (недавно созданный **sda5**), выполните следующую команду:

```
mkfs -v -t ext4 /dev/sda5
```

Замените sda5 именем вашего раздела LFS, если он отличается

```
alisa@LFS:~$ sudo mkfs -v -t ext4 /dev/sda2
[sudo] password for alisa:
mke2fs 1.46.5 (30-Dec-2021)
fs_types for mke2fs.conf resolution: 'ext4'
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
6111232 inodes, 24413952 blocks
1220697 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2172649472
746 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Filesystem UUID: 6fa5f487-0b91-4d73-8dd8-ec7193952d87
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

alisa@LFS:~$
```

Проверяем созданную нами файловую систему на новом разделе

```
parted -l
```

```
root@test:~# parted -l
Model: Msft Virtual Disk (scsi)
Disk /dev/sda: 215GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name  Flags
 1      1049kB  75,0GB  75,0GB  ext4
 2      75,0GB  77,0GB  2048MB  ext4
 3      77,0GB  81,1GB  4048MB  linux-swap(v1)        swap
 4      81,1GB  82,1GB  1024MB  fat32                 boot, esp
 5      82,1GB  189GB   107GB   ext4


root@test:~#
```

Проверим наличие swap-файла. У нас это созданный при установке sda3, который мы недавно запоминали.

```
swapon --show
```

```
root@test:~# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sda3 partition 3,8G   0B   -2
root@test:~#
```

Если вы используете существующий раздел подкачки, нет необходимости его форматировать. Если был создан новый раздел подкачки, его нужно будет инициализировать с помощью этой команды:

```
mkswap /dev/<yyy>
```

Замените <yyy> именем раздела подкачки.

2.6. Установка переменной $LFS

В этой книге переменная окружения LFS будет использоваться несколько раз. Вы должны убедиться, что эта переменная всегда определена в процессе сборки LFS. Она должна быть установлена на каталог, в котором вы будете создавать свою систему LFS — мы, для примера, будем использовать /mnt/lfs, но вы можете выбрать любой другой. Если вы собираете LFS на отдельном разделе, этот каталог будет точкой монтирования для раздела. Выберите расположение каталога и установите переменную с помощью следующей команды. Таким образом $LFS будет автоматически восприниматься системой как /mnt/lfs. Это достигается командой

```
export LFS=/mnt/lfs
```

```
root@test:~# export LFS=/mnt/lfs
root@test:~#
```

Внимание
Не забывайте проверять, что переменная LFS установлена, всякий раз, когда вы покидаете и снова входите в текущую рабочую среду (например, когда выполняете su для root или другого пользователя). Убедитесь, что переменная LFS настроена правильно:

```
echo $LFS
```

Убедитесь, что в выходных данных указан путь к местоположению сборки вашей системы LFS, то есть /mnt/lfs, если вы следовали примеру. Если вывод неверен, используйте команду, указанную ранее, чтобы установить $LFS в правильное значение каталога LFS.

```
root@test:~# echo $LFS
/mnt/lfs
root@test:~#
```

Примечание
Один из способов гарантировать, что переменная LFS всегда установлена, — отредактировать файл .bash_profile как в вашем личном домашнем каталоге, так и в /root/.bash_profile и добавить приведенную выше команду экспорта. Кроме того, оболочка, указанная в файле /etc/passwd для всех пользователей, которым нужна переменная LFS, должна быть bash, чтобы гарантировать, что файл /root/.bash_profile используется как часть процесса входа в систему.

Еще один способ, который используется для входа в хост-систему. При входе в систему через диспетчер графического дисплея пользовательский .bash_profile не используется при запуске виртуального терминала. В этом случае добавьте команду экспорта в файл .bashrc для своего пользователя и root. Кроме того, некоторые дистрибутивы используют тест «if» и не запускают оставшиеся инструкции .bashrc для не интерактивного вызова bash. Обязательно разместите команду экспорта перед тестом для не интерактивного использования.

2.7. Монтирование нового раздела

Теперь, когда файловая система создана, раздел должен быть смонтирован, чтобы хост-система могла получить доступ к нему. В книге предполагается, что файловая система монтируется в каталог, указанный в переменной LFS, описанной в предыдущем разделе.

Строго говоря, нельзя «смонтировать раздел». Монтируется файловая система на этом разделе. Но так как один раздел не может содержать несколько файловых систем, люди часто говорят о разделе и связанной с ним файловой системе так, как если бы они были одним и тем же.

Монтируем раздел командами (в нашем случае это раздел sda5, хотя, у вас может быть иначе)

```
mkdir -v $LFS
```

```
root@test:~# mkdir -v $LFS
mkdir: created directory '/mnt/lfs'
root@test:~#
```

```
mount -v -t ext4 /dev/sda5 $LFS
```

Замените **sda5** на имя вашего раздела **LFS** в случае необходимости.

```
root@test:~# mount -v -t ext4 /dev/sda5 $LFS
mount: /dev/sda5 mounted on /mnt/lfs.
root@test:~#
```

На выходе имеем:mount: /dev/sda5 смонтирован в /mnt/lfs.
Проверим наш смонтированный раздел

```
lsblk
```

```
root@test:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0  61,9M  1 loop /snap/core20/1405
loop2    7:2    0  63,9M  1 loop /snap/core20/2318
loop3    7:3    0 155,6M  1 loop /snap/firefox/1232
loop4    7:4    0 349,7M  1 loop /snap/gnome-3-38-2004/143
loop5    7:5    0 248,8M  1 loop /snap/gnome-3-38-2004/99
loop6    7:6    0  81,3M  1 loop /snap/gtk-common-themes/1534
loop7    7:7    0  91,7M  1 loop /snap/gtk-common-themes/1535
loop8    7:8    0  45,9M  1 loop /snap/snap-store/575
loop9    7:9    0  38,8M  1 loop /snap/snapd/21759
loop10   7:10   0   284K  1 loop /snap/snapd-desktop-integration/10
sda      8:0    0   200G  0 disk
├─sda1   8:1    0  69,8G  0 part /
├─sda2   8:2    0   1,9G  0 part /boot
├─sda3   8:3    0   3,8G  0 part [SWAP]
├─sda4   8:4    0   977M  0 part /boot/efi
└─sda5   8:5    0   100G  0 part /mnt/lfs
sr0     11:0    1  1024M  0 rom
root@test:~#
```

Если вы используете несколько разделов для LFS (например, один для /, а другой для /home), смонтируйте их вот так:

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
mkdir -v $LFS/home
mount -v -t ext4 /dev/<yyy> $LFS/home
```

Замените <xxx> и <yyy> соответствующими именами разделов.

Убедитесь, что этот новый раздел не смонтирован со слишком строгими разрешениями (такими как параметры **nosuid** или **nodev**). Запустите команду **mount** без каких-либо параметров, чтобы увидеть, какие параметры установлены для смонтированного раздела LFS. Если установлены nosuid и/или nodev, раздел должен быть размонтирован и смонтирован повторно.

```
mount
```

Проверим разрешения

```
/dev/sda5 on /mnt/lfs type ext4 (rw,relatime)
root@test:~#
```

Предупреждение

Приведенные выше инструкции предполагают, что вы не будете перезагружать компьютер в процессе сборки LFS. Если вы выключите свою систему, вам придется либо перемонтировать раздел LFS каждый раз, когда вы перезапускаете процесс сборки, либо изменить файл /etc/fstab вашей хост-системы, чтобы он автоматически монтировал его при загрузке. Например, вы можете добавить эту строку в свой /etc/fstab:

```
/dev/<xxx>  /mnt/lfs ext4    defaults      1     1
```

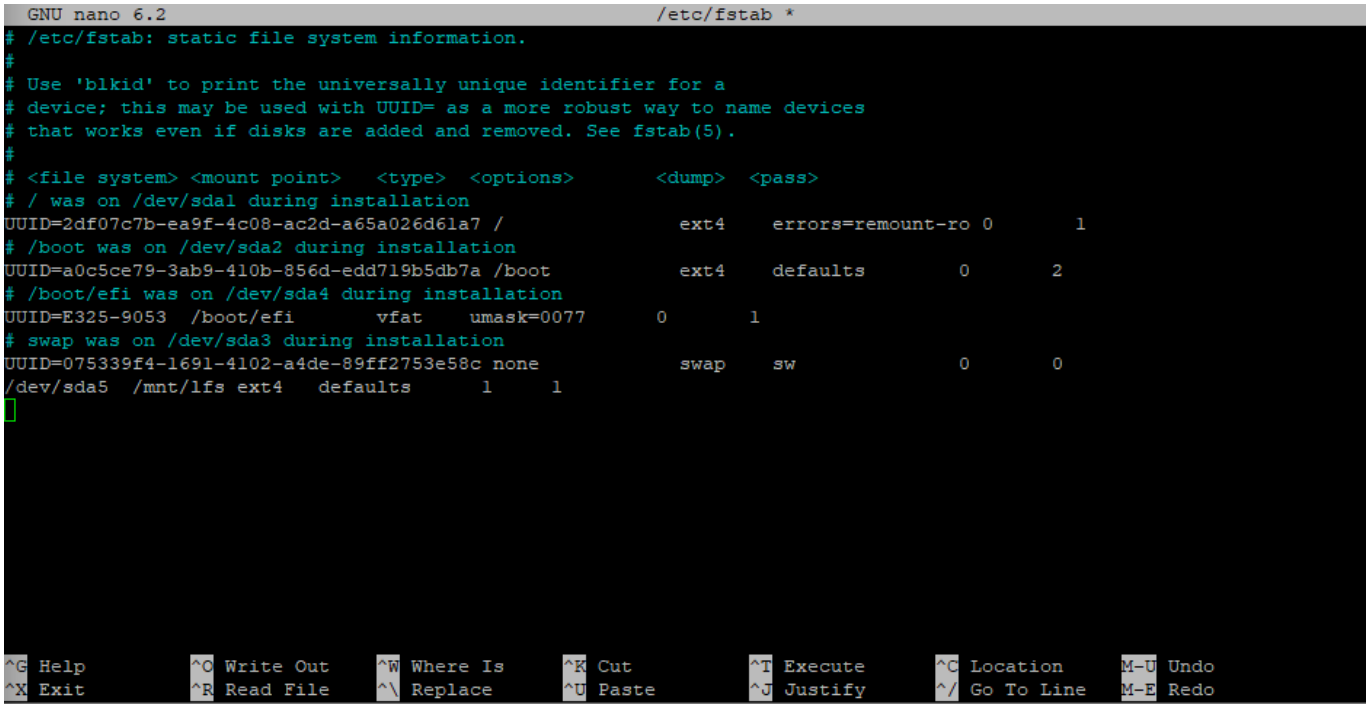
Если вы используете дополнительные разделы, обязательно добавьте их.

Редактируем файл /etc/fstab в редакторе nano

```
nano /etc/fstab
```

Добавим строку для автоматического монтажа раздела sda5

```
/dev/sda5  /mnt/lfs ext4   defaults      1     1
```



CTRL+O → ENTER → CTRL+X (Сохранить и выйти)

Если вы используете раздел подкачки, убедитесь, что он включен с помощью команды swapon:

```
/sbin/swapon -v /dev/sda3
```

Замените sda3 именем раздела подкачки.

Теперь, когда новый раздел LFS готов к работе, пришло время загрузить пакеты.

- След. chapter03