← 10.2 Создание файла /etc/fstab 10.4 Использование GRUB для настройки процесса загрузки →

10.3.Linux-6.13.2

Пакет Linux содержит ядро Linux.	
Приблизительное время сборки:	2.5 SBU
Требуемое дисковое пространство:	2.3 GB

10.3.1 Установка ядра

Сборка ядра включает несколько шагов — настройка, компиляция и установка. Прочитайте READMEфайл в исходном дереве ядра для альтернативных методов, которые отличают эту книгу от настройки ядра.

> **Важный**
> Первая сборка ядра Linux — одна из самых сложных задач в LFS. Правильное выполнение зависит от конкретного оборудования целевой системы и ваших конкретных потребностей. Для ядра доступно почти 12 000 элементов конфигурации, хотя для большинства компьютеров требуется лишь около трети из них. Редакторы LFS рекомендуют пользователям, не знакомым с этим процессом, достаточно внимательно следовать процедурам, описанным ниже. Цель состоит в том, чтобы довести начальную систему до точки, в которой вы сможете войти в командную строку при перезагрузке позже в Разделе 11.3, «Перезагрузка системы». На этом этапе оптимизация и настройка не являются целью.
>
> Для получения общей информации о настройке ядра см. https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt . Дополнительную информацию о настройке и сборке ядра можно найти по адресу https://anduin.linuxfromscratch.org/LFS/kernel-nutshell/ . Эти ссылки немного устарели, но все еще дают разумный обзор процесса.
>
> Если все остальное не помогает, вы можете обратиться за помощью в список рассылки lfs-support. Обратите внимание, что подписка обязательна, чтобы избежать спама.

Подготовьтесь к компиляции, выполнив следующую команду:

```
make mrproper
```

Это гарантирует, что дерево ядра абсолютно чистое. Команда ядра рекомендует выполнять эту команду перед каждой компиляцией ядра. Не полагайтесь на то, что исходное дерево будет чистым после распаковки.

Существует несколько способов настройки параметров ядра. Обычно это делается через интерфейс на основе меню, например:

```
make menuconfig
```

Значение необязательных переменных окружения make:

LANG=<host_LANG_value> LC_ALL=
Это устанавливает настройку локали на ту, которая используется на хосте. Это может быть необходимо для правильного рисования линий интерфейса menuconfig ncurses на текстовой консоли UTF-8 linux.

Если используется, обязательно замените <host_LANG_value> на значение переменной $LANG из вашего хоста. Вы также можете использовать вместо этого значение хоста $LC_ALLили $LC_CTYPE.

make menuconfig
Это запускает интерфейс ncurses, управляемый меню. Для других (графических) интерфейсов введите make help .

Примечание
Хорошей отправной точкой для настройки конфигурации ядра является запуск make defconfig . Это установит базовую конфигурацию в хорошее состояние, которое учитывает вашу текущую архитектуру системы.

Обязательно включите/отключите/настройте следующие функции, иначе система может работать некорректно или вообще не загрузиться:

```
General setup --->
   [ ] Compile the kernel with warnings as errors
[WERROR]
  CPU/Task time and stats accounting --->
        [*] Pressure stall information tracking
[PSI]
    [ ]  Require boot parameter to enable pressure stall
information tracking
                                                    ...
[PSI_DEFAULT_DISABLED]
  < > Enable kernel headers through /sys/kernel/kheaders.tar.xz
[IKHEADERS]
         [*]    Control    Group    support    --->
[CGROUPS]
                 [*]            Memory    controller
[MEMCG]
               [    /*]    CPU    controller    --->
[CGROUP_SCHED]
     # This may cause some systemd features malfunction:
           [ ] Group scheduling for SCHED_RR/FIFO
```

```
[RT_GROUP_SCHED]
  [ ] Configure standard kernel features (expert users) --->
[EXPERT]

Processor type and features --->
         [*]     Build     a     relocatable     kernel
[RELOCATABLE]
  [*]   Randomize the address of the kernel image (KASLR)
[RANDOMIZE_BASE]

General architecture-dependent options --->
    [*]  Stack  Protector  buffer  overflow  detection
[STACKPROTECTOR]
         [*]              Strong     Stack     Protector
[STACKPROTECTOR_STRONG]

[*]           Networking           support           --->
[NET]
  Networking options --->
                          [*]      TCP/IP      networking
[INET]
            <*>          The    IPv6    protocol    --->
[IPV6]

Device Drivers --->
  Generic Driver Options --->
              [    ]   Support   for   uevent   helper
[UEVENT_HELPER]
     [*] Maintain a devtmpfs filesystem to mount at /dev
[DEVTMPFS]
    [*]   Automount devtmpfs at /dev, after the kernel mounted
the rootfs
                                                          ...
[DEVTMPFS_MOUNT]
    Firmware loader --->
                <   /*>   Firmware   loading   facility
[FW_LOADER]
      [ ]    Enable the firmware sysfs fallback mechanism
                                                          ...
[FW_LOADER_USER_HELPER]
  Firmware Drivers --->
     [*] Export DMI identification via sysfs to userspace
[DMIID]
     [*] Mark VGA/VBE/EFI FB as generic system framebuffer
[SYSFB_SIMPLEFB]
  Graphics support --->
    <*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI
support) --->
...  [DRM]
      [*]    Display a user-friendly message when a kernel panic
occurs
```

```
...  [DRM_PANIC]
               (kmsg)        Panic   screen   formatter
[DRM_PANIC_SCREEN]
      [*]    Enable legacy fbdev support for your modesetting
driver
                                                          ...
[DRM_FBDEV_EMULATION]
              <*>          Simple   framebuffer   driver
[DRM_SIMPLEDRM]
    Console display driver support --->
                  [*]   Framebuffer   Console   support
[FRAMEBUFFER_CONSOLE]

File systems --->
        [*]    Inotify    support    for    userspace
[INOTIFY_USER]
  Pseudo filesystems --->
    [*] Tmpfs virtual memory file system support (former shm fs)
[TMPFS]
        [*]      Tmpfs  POSIX  Access  Control  Lists
[TMPFS_POSIX_ACL]
```

Включите некоторые дополнительные функции, если вы создаете 64-битную систему. Если вы используете menuconfig, включите их в порядке *CONFIG_PCI_MSI* сначала, затем *CONFIG_IRQ_REMAP*, и в конце, *CONFIG_X86_X2APIC* потому что опция отображается только после выбора ее зависимостей.

```
Processor type and features --->
                    [*]          Support          x2apic
[X86_X2APIC]

Device Drivers --->
                [*]        PCI        support        --->
[PCI]
     [*] Message Signaled Interrupts (MSI and MSI-X)
[PCI_MSI]
         [*]    IOMMU    Hardware    Support    --->
[IOMMU_SUPPORT]
           [*]   Support   for   Interrupt   Remapping
[IRQ_REMAP]
```

Если вы создаете 32-разрядную систему, работающую на оборудовании с оперативной памятью более 4 ГБ, измените конфигурацию так, чтобы ядро могло использовать до 64 ГБ физической оперативной памяти:

```
Processor type and features --->
  High Memory Support --->
                                          (X)          64GB
[HIGHMEM64G]
```

Если раздел для системы LFS находится на SSD-накопителе NVME (т.е. узел устройства для раздела — */dev/nvme** вместо */dev/sd**), включите поддержку NVME, иначе система LFS не загрузится:

```
Device Drivers --->
  NVME Support --->
                <*>    NVM    Express    block    device
[BLK_DEV_NVME]
```

Примечание
Хотя «Протокол IPv6» не является строго обязательным, его использование настоятельно рекомендуется разработчиками systemd.

Есть несколько других опций, которые могут быть желательны в зависимости от требований к системе. Список опций, необходимых для пакетов BLFS, см. в BLFS Index of Kernel Settings.

Примечание
Если ваше хостовое оборудование использует UEFI и вы хотите загрузить систему LFS с его помощью, вам следует настроить некоторые параметры ядра, следуя инструкциям на странице BLFS, **даже если вы будете использовать загрузчик UEFI из хостового дистрибутива .**

Обоснование вышеуказанных элементов конфигурации:

Randomize the address of the kernel image (KASLR)
Включите ASLR для образа ядра, чтобы смягчить некоторые атаки, основанные на фиксированных адресах конфиденциальных данных или кода в ядре.

Compile the kernel with warnings as errors
Это может привести к сбою сборки, если компилятор и/или конфигурация отличаются от тех, что были у разработчиков ядра.

Enable kernel headers through /sys/kernel/kheaders.tar.xz
Для этого потребуется cpio для сборки ядра. cpio не устанавливается LFS.

Configure standard kernel features (expert users)
Это заставит некоторые опции отображаться в интерфейсе конфигурации, но изменение этих опций может быть опасным. Не используйте это, если вы не знаете, что делаете.

Strong Stack Protector
Включить SSP для ядра. Мы включили его для всего пользовательского пространства с помощью –enable-default-ssp настройки GCC, но ядро не использует настройку GCC по умолчанию для SSP. Мы включаем его явно здесь.

Support for uevent helper
Установка этого параметра может помешать управлению устройством при использовании Udev.

Maintain a devtmpfs
Это создаст автоматизированные узлы устройств, которые заполняются ядром, даже без запущенного Udev. Затем Udev работает поверх этого, управляя разрешениями и добавляя символические ссылки. Этот элемент конфигурации требуется для всех пользователей Udev.

Automount devtmpfs at/dev
Это позволит смонтировать представление ядра устройств в /dev при переключении на корневую файловую систему непосредственно перед запуском init.

Display a user-friendly message when a kernel panic occurs
Это заставит ядро правильно отображать сообщение в случае возникновения паники ядра, а работающий драйвер DRM поддерживает это. Без этого было бы сложнее диагностировать панику: если драйвер DRM не запущен, мы бы находились на консоли VGA, которая может содержать только 24 строки, и соответствующее сообщение ядра часто сбрасывается; если драйвер DRM запущен, отображение часто полностью портится при панике. Начиная с Linux-6.12, ни один из выделенных драйверов для основных моделей GPU на самом деле не поддерживает это, но это поддерживается « Простым драйвером буфера кадра » , который работает на буфере кадра VESA (или EFI) до загрузки выделенного драйвера GPU. Если выделенный драйвер графического процессора собран как модуль (а не часть образа ядра) и `initramfs` не используется, эта функциональность будет работать нормально до монтирования корневой файловой системы, и ее уже достаточно для предоставления информации о большинстве ошибок конфигурации LFS, вызывающих панику (например, неправильная root=настройка в Разделе 10.4, «Использование GRUB для настройки процесса загрузки»).

Panic screen formatter
Установите это kmsg, чтобы убедиться, что последние строки сообщений ядра отображаются при возникновении паники ядра. Значение по умолчанию, user, заставит ядро отображать только « дружественное пользователю » сообщение о панике, которое бесполезно при диагностике. Третий вариант, `qr_code`, заставит ядро сжать последние строки сообщений ядра в QR-код и отобразить его. QR-код может содержать больше строк сообщений, чем обычный текст, и его можно декодировать с помощью внешнего устройства (например, смартфона). Но для этого требуется компилятор Rust, который LFS не предоставляет.

Mark VGA/VBE/EFI FB as generic system framebuffer иSimple framebuffer driver
Они позволяют использовать буфер кадра VESA (или буфер кадра EFI, если загрузка системы LFS осуществляется через UEFI) в качестве устройства DRM. Буфер кадра VESA будет настроен GRUB (или буфер кадра EFI будет настроен прошивкой UEFI), поэтому обработчик паники DRM может работать до загрузки драйвера DRM, специфичного для GPU.

Enable legacy fbdev support for your modesetting driverиFramebuffer Console support
Они необходимы для отображения консоли Linux на GPU, управляемом драйвером DRI (Direct Rendering Infrastructure). Поскольку CONFIG_DRM (Direct Rendering Manager) включен, мы должны включить эти две опции, иначе мы увидим пустой экран после загрузки драйвера DRI.

Support x2apic
Поддержка работы контроллера прерываний 64-битных процессоров x86 в режиме x2APIC. x2APIC может быть включен прошивкой на 64-битных системах x86, и ядро без этой опции будет паниковать при загрузке, если x2APIC включен прошивкой. Эта опция не имеет никакого эффекта, но и не навредит, если x2APIC отключен прошивкой.

В качестве альтернативы, `make oldconfig` может быть более подходящим в некоторых ситуациях. Смотрите READMEфайл для получения дополнительной информации.

При желании пропустите настройку ядра, скопировав файл конфигурации ядра, `.config`, из хост-системы (предполагая, что он доступен) в распакованный `linux-6.13.2` каталог. Однако мы не рекомендуем этот вариант. Часто лучше изучить все меню конфигурации и создать конфигурацию ядра с нуля.

Скомпилируйте образ ядра и модули:

```
make
```

При использовании модулей ядра может потребоваться настройка модуля /etc/modprobe.d. Информация, касающаяся модулей и конфигурации ядра, находится в разделе 9.3 «Обзор обработки устройств и модулей» и в документации ядра в linux-6.13.2/Documentationкаталоге. Также может быть интересен modprobe.d(5).

Если поддержка модулей не отключена в конфигурации ядра, установите модули с помощью:

```
make modules_install
```

После завершения компиляции ядра требуются дополнительные шаги для завершения установки. Некоторые файлы необходимо скопировать в `/boot` каталог.

> **Осторожность**
> Если вы решили использовать отдельный `/boot` раздел для системы LFS (возможно, разделяя `/boot` раздел с дистрибутивом хоста), скопированные ниже файлы должны быть там. Самый простой способ сделать это — `/boot` сначала создать запись для `/etc/fstab` (подробности в предыдущем разделе), а затем выполнить следующую команду как rootпользователь в среде `chroot`:
>
> ```
> mount /boot
> ```
>
> Путь к узлу устройства в команде опущен, поскольку mount может прочитать его из */etc/fstab.*

Путь к образу ядра может различаться в зависимости от используемой платформы. Имя файла ниже можно изменить по своему вкусу, но основа имени файла должна быть vmlinuz , чтобы быть совместимой с автоматической настройкой процесса загрузки, описанной в следующем разделе. Следующая команда предполагает архитектуру x86:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-6.13.2-lfs-12.3-systemd-rc1
```

`System.map` это файл символов для ядра. Он отображает точки входа каждой функции в API ядра, а также адреса структур данных ядра для работающего ядра. Он используется как ресурс при исследовании проблем ядра. Выполните следующую команду для установки файла карты:

```
cp -iv System.map /boot/System.map-6.13.2
```

Файл конфигурации ядра, `.config`c озданный на шаге `make menuconfig` выше, содержит все параметры конфигурации для только что скомпилированного ядра. Рекомендуется сохранить этот файл для дальнейшего использования:

```
cp -iv .config /boot/config-6.13.2
```

Установите документацию для ядра Linux:

```
cp -r Documentation -T /usr/share/doc/linux-6.13.2
```

Важно отметить, что файлы в исходном каталоге ядра не принадлежат root . Всякий раз, когда пакет распаковывается как пользователь `root` (как мы делали внутри `chroot`), файлы имеют идентификаторы пользователя и группы, которые были на компьютере упаковщика. Обычно это не является проблемой для установки любого другого пакета, поскольку исходное дерево удаляется после установки. Однако исходное дерево Linux часто сохраняется в течение длительного времени. Из-за этого существует вероятность того, что любой идентификатор пользователя, который использовал упаковщик, будет назначен кому-то на машине. Этот человек затем получит доступ на запись в исходный код ядра.

Примечание
Во многих случаях конфигурацию ядра необходимо будет обновить для пакетов, которые будут установлены позже в BLFS. В отличие от других пакетов, нет необходимости удалять исходное дерево ядра после установки нового ядра.

Если планируется сохранить исходное дерево ядра, запустите chown -R 0:0 в `linux-6.13.2` каталоге, чтобы убедиться, что все файлы принадлежат пользователю `root`.

Если вы обновляете конфигурацию и перестраиваете ядро из сохраненного исходного дерева ядра, обычно вам не следует запускать команду `make mrproper`. Команда очистит `.config` файл и все .o файлы из предыдущей сборки. Несмотря на то, что восстановление `.config` из копии в легко `/boot`, очистка всех .o файлов все равно является пустой тратой: для простого изменения конфигурации часто .отребуется (пере)собрать только несколько файлов, а система сборки ядра правильно пропустит другие .o файлы, если они не будут очищены.

С другой стороны, если вы обновили GCC, вам следует запустить `make clean`, чтобы удалить все .офайлы из предыдущей сборки, иначе новая сборка может завершиться ошибкой.

Предупреждение
В некоторых документах ядра рекомендуется создать символическую ссылку, `/usr/src/linux` указывающую на исходный каталог ядра. Это относится к ядрам до серии 2.6 и **не должно** создаваться в системе LFS, поскольку это

может вызвать проблемы для пакетов, которые вы захотите собрать после завершения базовой системы LFS.

10.3.2 Настройка порядка загрузки модулей Linux

Большую часть времени модули Linux загружаются автоматически, но иногда требуется определенное направление. Программа, которая загружает модули, modprobe или insmod, использует /etc/modprobe.d/usb.conf для этой цели. Этот файл необходимо создать, чтобы, если драйверы USB (ehci_hcd, ohci_hcd и uhci_hcd) были собраны как модули, они загружались в правильном порядке; ehci_hcd необходимо загрузить до ohci_hcd и uhci_hcd, чтобы избежать вывода предупреждения во время загрузки.

Создайте новый файл /etc/modprobe.d/usb.conf, выполнив следующую команду:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
EOF
```

10.3.3 Содержимое Linux

Установленные файлы:	config-6.13.2, vmlinuz-6.13.2-lfs-12.3-systemd-rc1 и System.map-6.13.2
Установленные каталоги:	/lib/modules, /usr/share/doc/linux-6.13.2

Краткие описания

config-6.13.2	Содержит все параметры конфигурации ядра.
vmlinuz-6.13.2-lfs-12.3-systemd-rc1	Движок системы Linux. При включении компьютера ядро — это первая часть операционной системы, которая загружается. Оно обнаруживает и инициализирует все компоненты оборудования компьютера, затем делает эти компоненты доступными в виде дерева файлов для программного обеспечения и превращает один процессор в многозадачную машину, способную запускать множество программ, казалось бы, одновременно.
System.map-6.13.2	Список адресов и символов; он отображает точки входа и адреса всех функций и структур данных в ядре.

← 10.2 Создание файла /etc/fstab 10.4 Использование GRUB для настройки процесса загрузки →