Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 1

По дисциплине «Операционные системы»

На тему «Знакомство с ОС: установка и аппаратная конфигурация»

Выполнила:

Студентка 3 курса 6 группы

Жучкевич Екатерина Сергеевна  
  
Дата выполнения: 11.09.2025

Минск 2025

# **ВВЕДЕНИЕ**

Целью работы является освоение процесса установки операционной системы, приобретение навыков базовой настройки и конфигурации. А также научиться создавать пользовательские учётные записи и подготовить рабочее место для выполнения типовых задач.

**Постановка задачи:**

1. Требуется установить операционную систему семейства Windows NT).
2. Требуется установить операционную систему семейства Linux.
3. Требуется найти информацию об аппаратной конфигурации устройств(-а) на которых(-ом) установлены операционные системы из заданий выше.

Для установки Linux From Scratch будет использоваться приложение для контеризации и создания виртуальных машин Oracle VirtualBox. Установка операционной системы будет производиться на базе ОС Gentoo.

# **1 Общая информация о системах**

# **1.1 Операционная система**

Операционная система: Linux From Scratch 12.2-systemd

Архитектура: x64

Дата установки: 7 сентября

Пользовательские учетные записи: на данный момент нет пользовательских учетных записей.



Рисунок 1.1 – Информация об установленной системе

# **1.2 Базовая конфигурация**

Имя компьютера: katusha

Рабочая группа: нет в линуксе

Сетевые настройки представлены на рисунке 1.2.

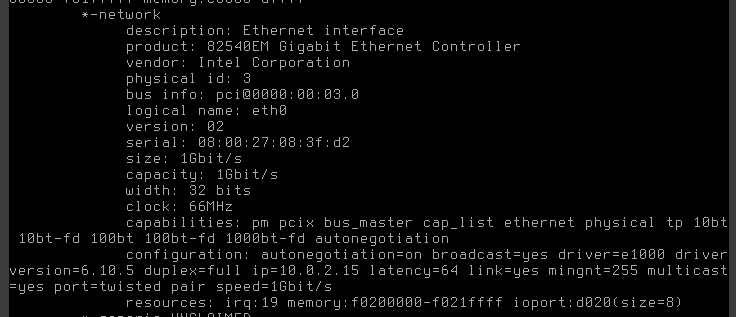


Рисунок 1.2 – Сетевые настройки

# **1.3 Центральный процессор**

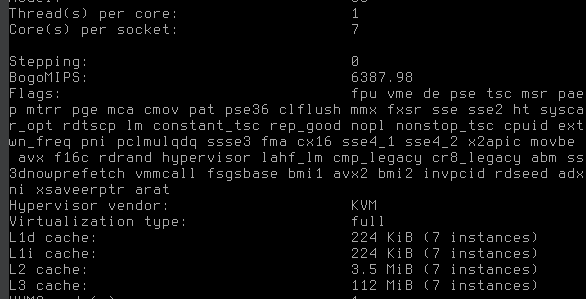
Центральный процессор: AMD Ryzen 7 5800H

Тактовая частота: нет данных

Количество ядер и потоков: 7 ядер, 7 потоков

Кэш-память: L1d – 224Кб, L1i – 224Кб, L2 – 3,5Мб, L3 – 112Мб

Технологический процесс: нет данных



# **1.4 Оперативная память**

Количество и наименования планок оперативной памяти: 1 планка.

Объем и тип памяти: 7950МБ

Номера используемых слотов: нет данных

Частота памяти: нет данных

Распределение по каналам: нет данных

Вся информация представлена на рисунке.

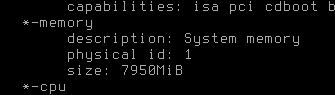


Рисунок – Информация об оперативной памяти

# **1.5 Материнская плата**

Производитель и модель: VirtualBox Oracle Corporation

Чипсет: нет данных

Версия BIOS: 1.2

Слоты расширения: нет данных

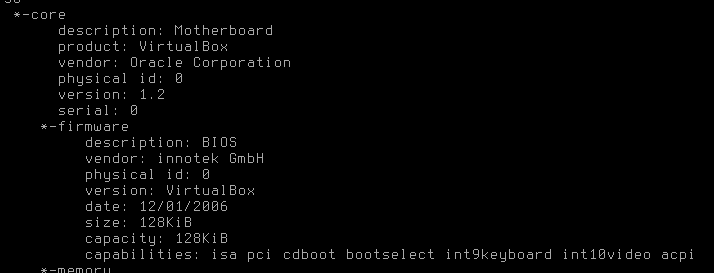


Рисунок – Информация о материнской плате

# **1.6 Графическая подсистема**

Модель видеокарты: SVGA II Adapter

Тип и объем памяти: нет данных

Версия драйверов: 00

Поддерживаемые технологии: vga\_controller

Информация о графической подсистеме представлена на рисунке

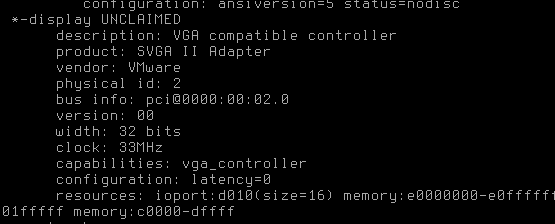


Рисунок – Информация о графической подсистеме

# **1.7 Накопители**

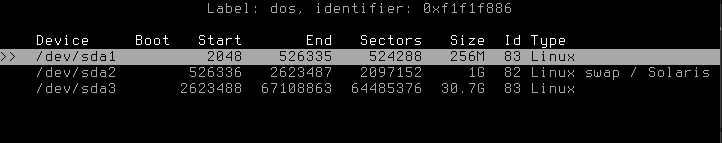
Тип и объем жестких дисков: ATA Disk, 34Гб

Файловая система: ext4

Свободное место: информация представлена на рисунке

Скорость вращения: нет данных







# **1.2 Периферийные устройства**

Наименование подключенных мониторов:

Разрешения мониторов:

Наименование видеоадаптеров:

Объем памяти видеоадаптеров:

Ширина шины памяти:

Наименование сетевых адаптеров:

Максимальная пропускная способность сетевых адаптеров:

# **2 Практическая часть**

Введение хзззззззззз в процессе разработки, также будет представлена архитектура системы.

# **2.1 Процесс установки**

# **2.1.1 Основные этапы установки Windows**

C# — современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он является основным языком для создания приложений на платформе .NET, что делает его популярным выбором для разработки различных типов программного обеспечения, включая веб-приложения, мобильные приложения и десктопные решения. C# предлагает высокую производительность благодаря компиляции в промежуточный язык. Язык имеет довольно простой и понятный синтаксис, что является огромным достоинством, упрощает написание и сопровождение кода. C# легко интегрируется с различными библиотеками и фреймворками, такими как ASP.NET и Entity Framework, что ускоряет разработку и улучшает функциональность.

# **2.1.1 Основные этапы установки Linux**

Для создания LFS требуется установить уже готовый дистрибутив UNIX-like системы. Мой выбор пал на ОС Gentoo для удобства.

Для начала нужно проверить, установлены пакеты, которые требуются для сборки. Команда, указанная на рисунке 1 выводит список файлов, которые находятся в текущем разделе.

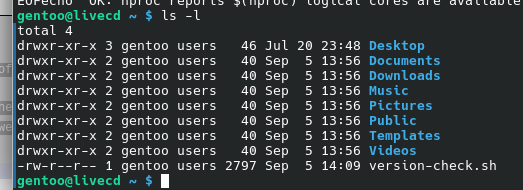


Рисунок 2.1.1.1 – Список файлов в текущем разделе

Далее используем команду bash с названием файла (рис. 2), который выведет все установленные пакеты и их версии.

.

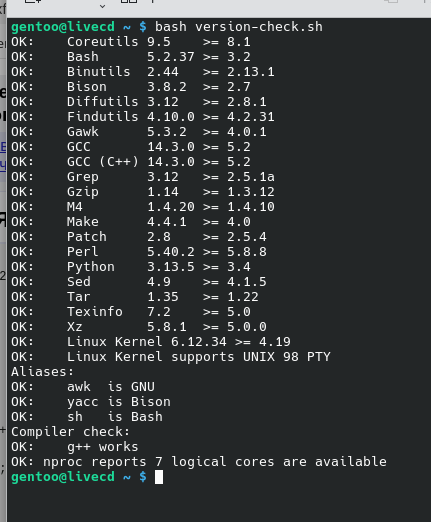


Рисунок 2.1.1.2 – Список установленных пакетов

Обращаю внимание, что версия ядра 4.19 (в книге этому уделен отдельный абзац), а также ядро хоста поддерживает псевдотерминал UNIX 98(PTY).

Версии пакетов совпадают.

Следующий шаг – создать разделы, которые будут выделены под LFS. Для начала нужно зайти с root’а.

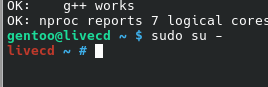


Рисунок 2.1.1.3 – Root-пользователь Gentoo

После загрузки с LiveCD мы вошли под root-пользователем. Теперь у нас есть права на выполнение системных команд. Следующим шагом будет обращение к справке о доступных дисках с помощью команды fdisk -l (флаг -l для вывода списком).

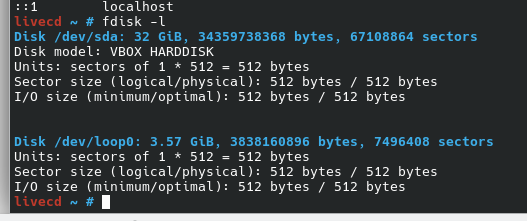


Рисунок 2.1.1.4 – Справка fdisk

Поскольку система работает в виртуальной машине, второй отображённый loop-диск нам не требуется. Нас интересует диск sda, обозначенный как VBOX HARDDISK.

# **2.1.1.1 Подготовка разделов**

Для начала выбираем диск, который нам нужен. Для просмотра конфигурации диска вводим команду p.

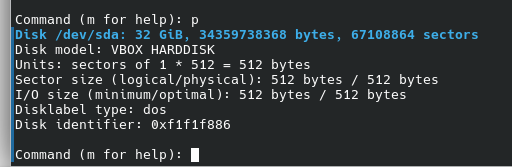


Рисунок 2.1.1.1.1 – Конфигурация диска

Далее создаем разделы.

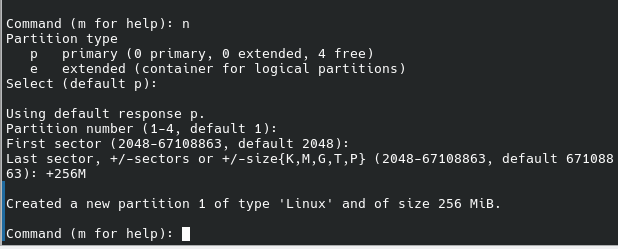


Рисунок 2.1.1.1.2 – Создание первого раздела

Мы создали первый стандартный (default) раздел на 256 мегабайт под boot. Аналогично создаем второй раздел на 1 гигабайт под swap и третий раздел. Повторяем все действия, только в последнем пункте не выделяем определенную память, а берем всю оставшуюся (click enter for default). Вышло 30.7Гб. Это будет раздел под всю систему линукс.

Теперь можно посмотреть все выделенные разделы.

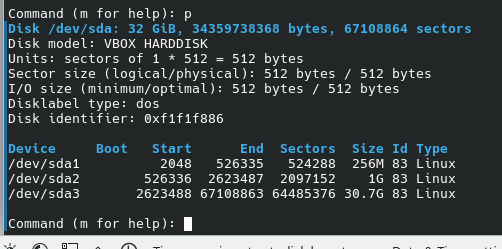


Рисунок 2.1.1.1.3 – Демонстрация созданных разделов

Необходимо изменить тип второго раздела: вместо «Linux» он должен быть «swap». Для смены типа используется команда t. Чтобы просмотреть список возможных ID разделов, применяется команда L. Для раздела под подкачку следует указать значение 82 (Linux swap).

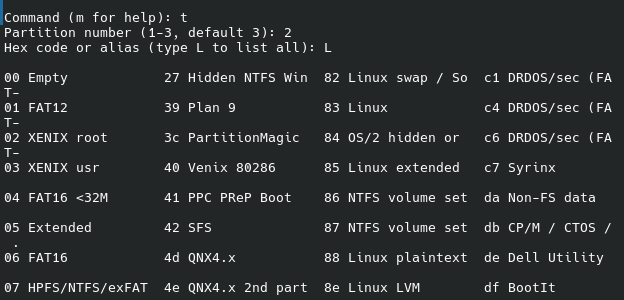


Рисунок 2.1.1.1.4 – Результат команды L

Чтобы убедиться в смене типа раздела, вызываем команду p.

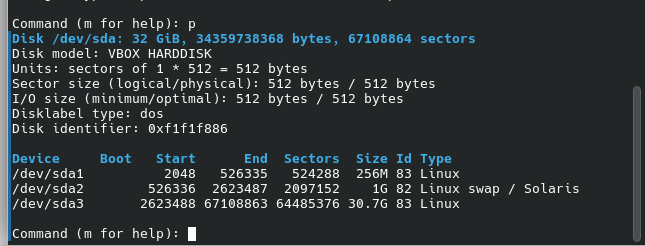


Рисунок 2.1.1.1.5 – Результат команды p

Далее нужно записать изменения командой w.

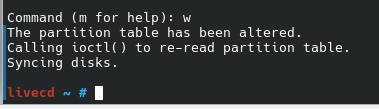


Рисунок 2.1.1.1.6 – Запись изменений

Следующим шагом является форматирование первого и третьего раздела. Для этого используется команда mkfs.

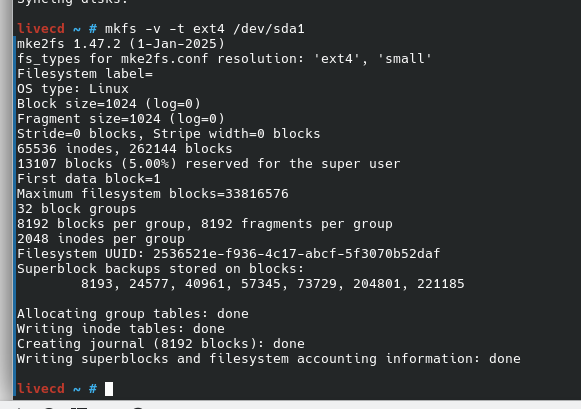


Рисунок 2.1.1.1.7 – Форматирование первого раздела

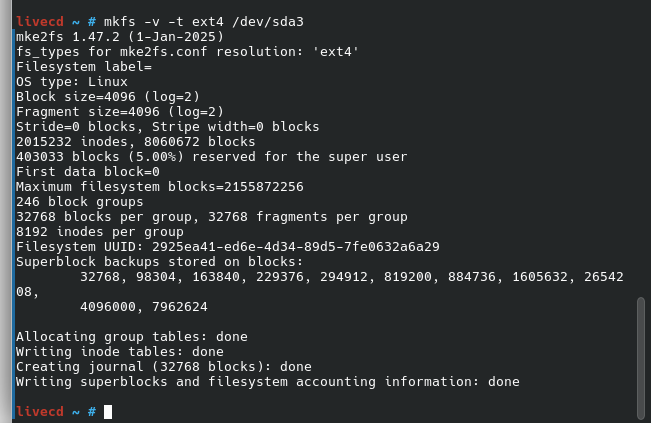


Рисунок 2.1.1.1.8 – Форматирование третьего раздела

Так как второй раздел предназначен для подкачки, для него используется отдельная команда mkswap.

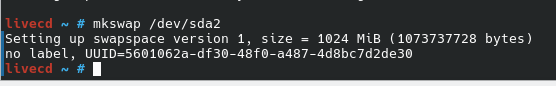


Рисунок 2.1.1.1.8 – Форматирование раздела подкачки

Следующим этапом является создание переменной LFS для того, чтобы все пакеты и конфигурации устанавливались не в текущую систему, а в отведённое пространство. Также монтируем разделы.

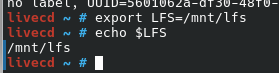


Рисунок 2.1.1.1.9 – Создание переменной окружения

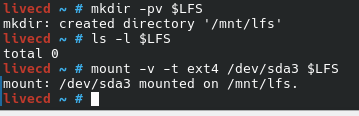


Рисунок 2.1.1.1.9 – Монтирование раздела будущей системы

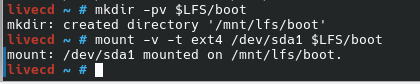


Рисунок 2.1.1.1.10 – Монтирование загрузочного раздела

И проверяем монтирование. Следует обратить внимание только на последние две строки. Также, так как мы создали раздел подкачки, нужно убедиться, что он также включен.

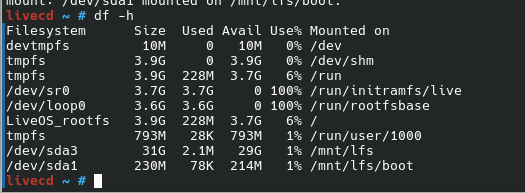


Рисунок 2.1.1.1.11 – Проверка монтирования

Для хранения архивов и патчей будет служить директорий sources. Сразу же сделаем из него директорий типа «sticky bit». Это означает, что даже если несколько пользователей имеют право на запись в каталог, только владелец файла может удалить файл в подобном каталоге.

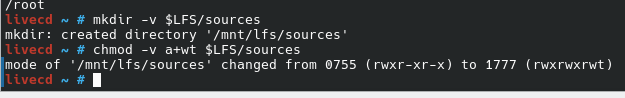


Рисунок 2.1.1.1.12 – Создание директория sources

Далее нужно скачать все необходимые пакеты и патчи. Обращаю внимание, что пакеты именно для systemd сборки. Более новой и быстрой init системы.

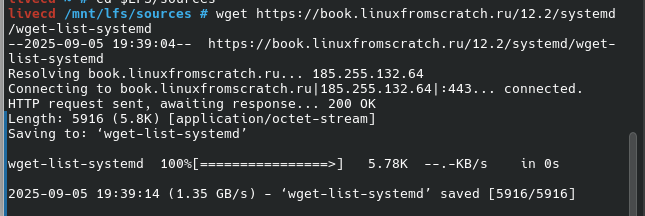


Рисунок 2.1.1.1.13 – Скачивание сборки

Чтобы проверить, правильно ли установлены все пакеты, скачиваем md5sum ссылкой из книги (рисунок 18).

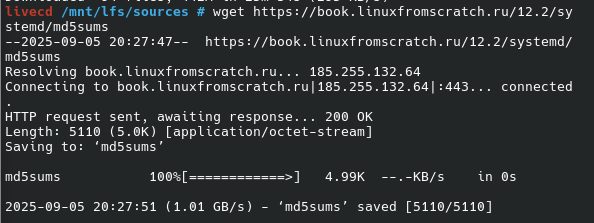


Рисунок 2.1.1.1.14 – Скачивание сборки

И проверяем командой pushd (рис. 19).

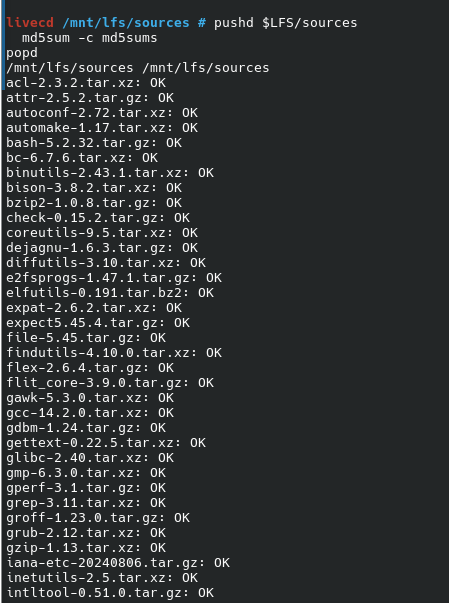


Рисунок 2.1.1.1.15 – Результат проверки сборки

Нужно присвоить указанные файлы имени root и создать необходимую иерархию каталогов.

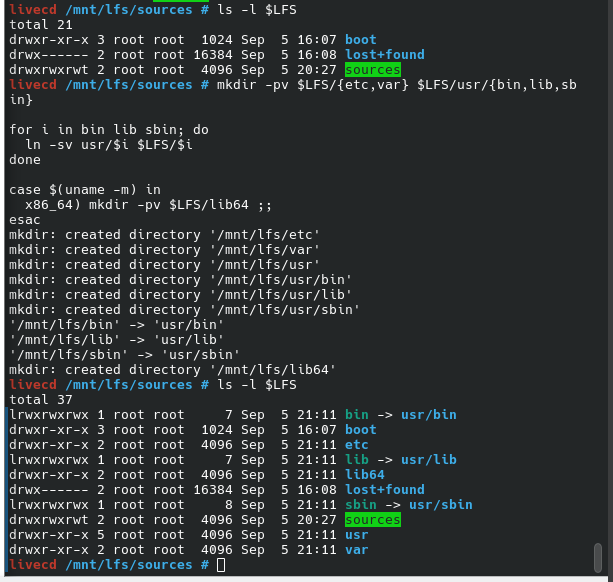


Рисунок 2.1.1.1.16 – Присваивание файлов и просмотр директория

Далее нужно создать отдельный каталог под кросс-компилятор. После этого работать под пользователем root становится опасно, поэтому следует создать непривилегированного пользователя.

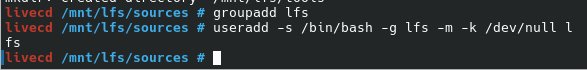


Рисунок 2.1.1.1.17 – Создание нового пользователя

Назначаем пользователя lfs владельцем, тем самым предоставляя ему доступ к каталогам папки $LFS.

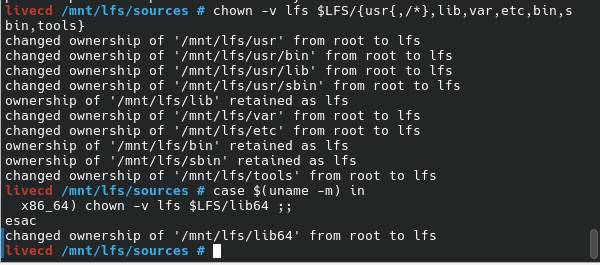


Рисунок 2.1.1.1.18 – Смена владельца

Для смены пользователя вводится команда sudo - lfs. Затем требуется создать два новых файла запуска для оболочки bash.

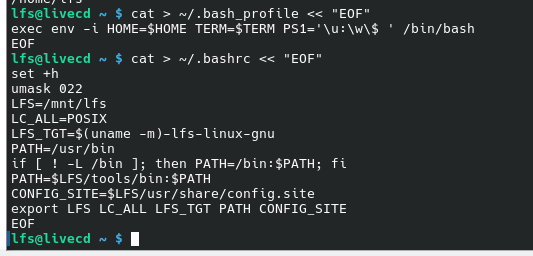


Рисунок 2.1.1.1.19 – Создание файлов

Далее устанавливаем количество ядер для быстроты. Чтобы не указывать это каждый раз, пропишем это количество прямо в файле .bashrc.

Переменная nproc – количество доступных ядер. Это число будет автоматически подставлено. Чтобы просто узнать количество доступных ядер, нужно ввести просто nproc. В моем случае, выводом станет цифра 7.

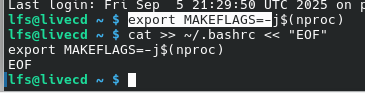


Рисунок 2.1.1.1.20 – Установка количества ядер

Далее проверим, все ли хорошо. Для этого обратимся к переменной MAKEFLAGS вместо того, чтобы искать определенную строку в файле .bash. Также, для демонстрации правильно использованной команды export, сменим пользователя на root, а затем снова вернемся в lfs. Это представлено на рисунке 25.

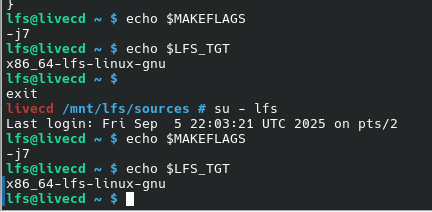


Рисунок 2.1.1.1.21 – Чтение переменной и смена пользователя

# **2.1.1.2 Установка пакетов**

Следует отметить, что в данном разделе не рассматривается установка каждого пакета по отдельности. Поскольку порядок действий во многом совпадает, будут приведены лишь некоторые из пакетов.

Первым пакетом является Binutils-2.43.1. Для начала нужно достать его из архива.



Рисунок 2.1.1.2.1 – Распаковка пакета

Некоторые из пакетов требуют создания дополнительного директория build.

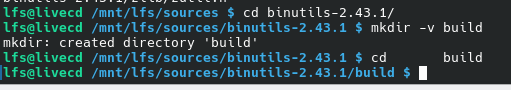


Рисунок 2.1.1.2.2 – Создание директория build

Для одновременной конфигурации, компиляции и установки пакета пропишем команды через логическое «и» (&&). Так как пакет Binutils является первым для установки, авторы учебного просят посчитать время его установки для будущего измерения времени установки других пакетов (SBU). Поэтому команду, описанную выше, следует обернуть в логический блок команды time.

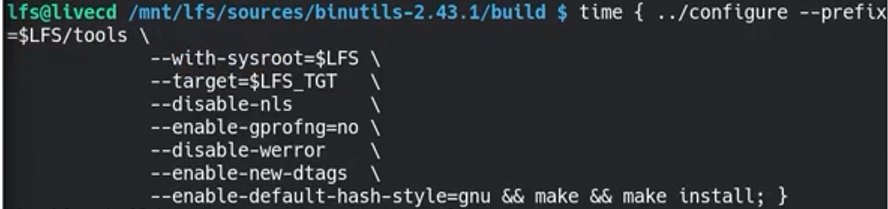


Рисунок 2.1.1.2.3 – Установка пакета Binutils

Время установки составило почти 1.2 минуты.



Рисунок 29 – Время установки пакета Binutils

Далее возвращаемся в директорий sources и удаляем оставшиеся после установки пакета файлы.

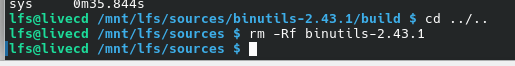


Рисунок 30 – Удаление оставшихся файлов

Устанавливаем пакет GCC-14.2.0. В документации GCC рекомендуется собирать его в отдельном каталоге. Компилируем пакет с помощью команды time make (можно просто make, time используется для подсчета итогового времени), далее устанавливаем пакет с помощью команды make install.

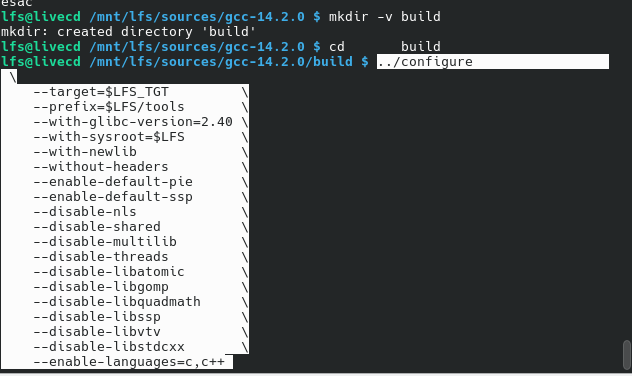


Рисунок 31 – Подготовка GCC к установке

Повторяем аналогичные действия ко всем пактам, описанных в текущей главе книги.

Для дальнейших действий нужно войти в окружение chroot. Для начала проверяем окружение командой echo $LFS, далее изменяем владельца каталогов $LFS/ на пользователя root.

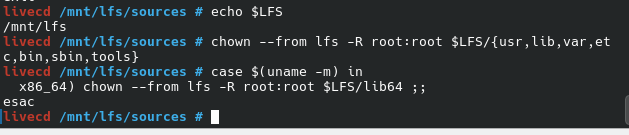


Рисунок 32 – Смена владельца каталогов

Далее создаем виртуальную файловую систему Kernel и монтируем её.

Затем входим в окружение Chroot и настраиваем его. Это понадобится для установки конечной системы.

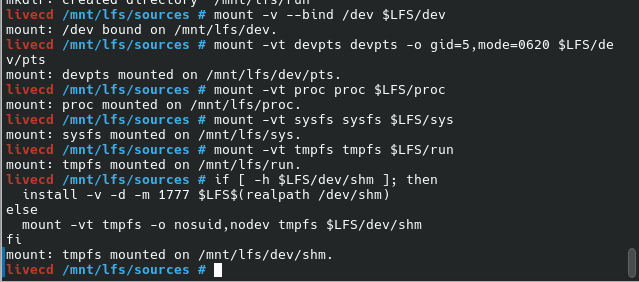


Рисунок 33 – Создание и монтирование Kernel

# **2.1.1.3 Системные настройки**

После установки базового системного программного обеспечения, необходимо начать процесс настройки и сборки ядра Linux. Главное меню конфигуратора ядра представлено на рисунке 34.

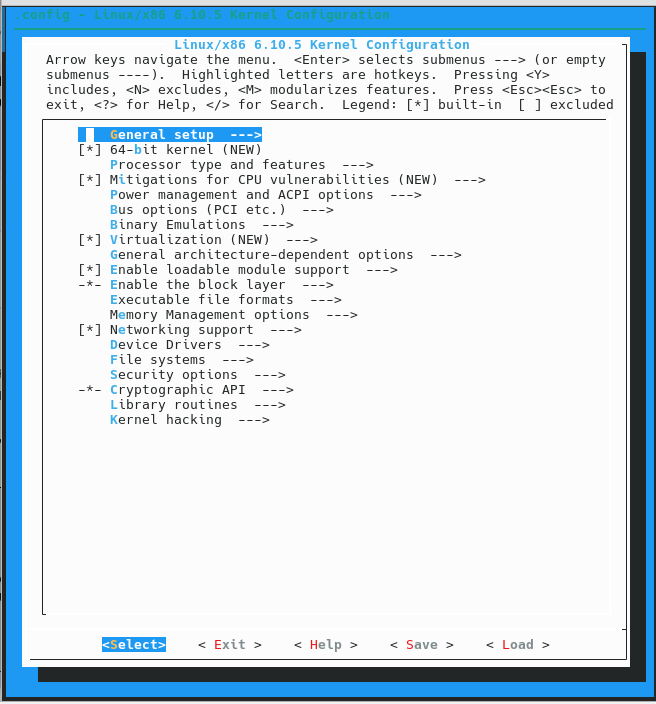


Рисунок 34 – Главное меню конфигуратора ядра

Все настройки должны быть сохранены в конфигурационный файл. Выбор файла для сохранения представлен на рисунке 35.

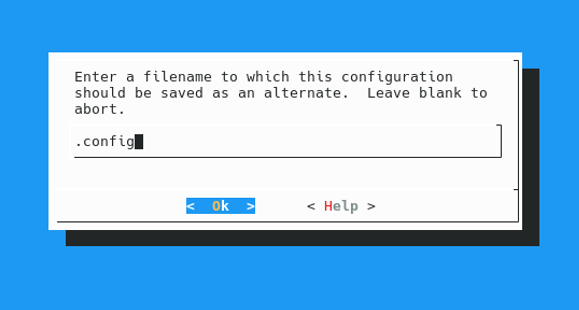


Рисунок 34 – Выбор файла для сохранения конфигурации ядра

Конфигурационный файл для загрузчика GRUB с добавлением LFS представлен на рисунке 35.

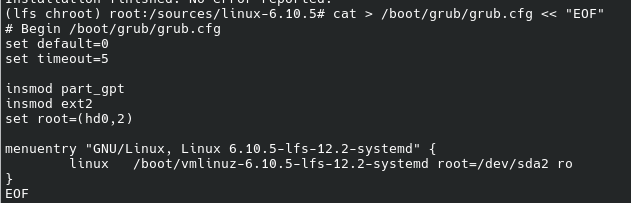


Рисунок 35 – Конфигурационный файл GRUB

Далее следует перезагрузить виртуальную машину. При новом включении нужно отключить образ Gentoo. Теперь система готова к работе. Экран входа представлен на рисунке 36.

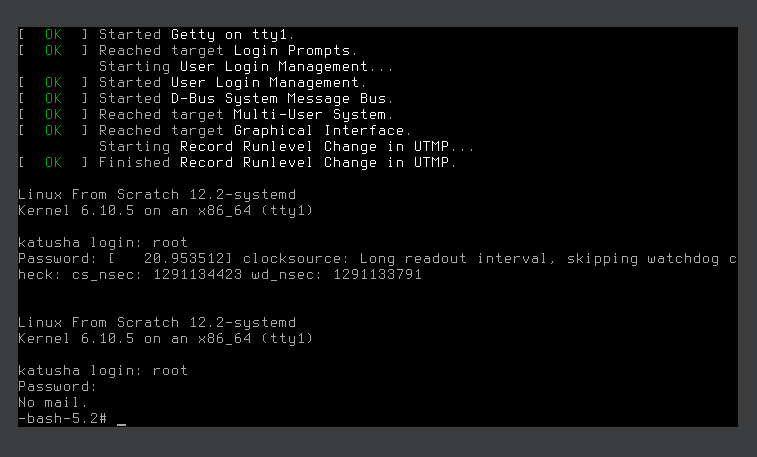


Рисунок 36 – Экран входа Linux From Scratch

Создание нового пользователя представлено на рисунке 52.

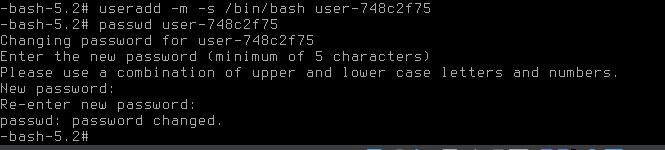


Рисунок – Создание нового пользователя

# **2.2 Схемы и диаграммы**

Диаграмма вариантов

# **ЗАКЛЮЧЕНИЕ**

Разработанное программное средство "Магазин автозапчастей" представляет собой значительный шаг вперед в сфере управления продажами и учетом автозапчастей. Данное приложение обеспечивает удобство, эффективность и простоту в организации процессов как для продавцов, так и для клиентов.

Целью разработки являлось создание инновационного программного решения, удовлетворяющего современные потребности пользователей в сфере автобизнеса. Основные принципы, заложенные в разработку, включают разнообразие функционала, а также интуитивно понятный интерфейс для легкого освоения пользователями. Это приложение может стать удобным инструментом в современной бизнес-среде, способствуя повышению эффективности работы профессионалов.

Одним из главных преимуществ программного средства является его простота использования. Дружественный пользовательский интерфейс и интуитивно понятные функции делают работу с приложением комфортной и без лишних сложностей. Пользователи смогут быстро освоиться с программой и использовать ее по назначению без необходимости дополнительного обучения.

В процессе реализации программного средства были достигнуты вышеуказанные требования и задачи. Применение паттерна MVVM обеспечило эффективное управление данными и логикой приложения, улучшенную архитектуру и обеспечило высокую отзывчивость и удобство использования пользовательского интерфейса.

Работа над проектом также позволила получить ценный опыт взаимодействия со сторонними библиотеками. Изучение и внедрение этих библиотек в проект позволило расширить его функциональность и улучшить его производительность. Разработанное программное средство "Магазин автозапчастей" является функциональным, удобным и привлекательным для пользователей.