МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ПРАКТИЧЕСКАЯ РАБОТА №1.1**

**по дисциплине: «Программирование»**

**на тему: «ОС Linux»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АИ-42», «АВТФ»,  *Костин Константин Евгеньевич*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc193887613)

[Ход выполнения 4](#_Toc193887614)

[**1. Процессы, память** 4](#_Toc193887615)

[**2. Утилиты управления** 8](#_Toc193887616)

[**3. Файлы, пользователи** 12](#_Toc193887617)

[**4. Текст** 15](#_Toc193887618)

[**5. Файлы** 16](#_Toc193887619)

[**6. Процесс сборки программы на C++** 16](#_Toc193887620)

[**7. Процесс сборки программы на Go.** 17](#_Toc193887621)

[Вывод 18](#_Toc193887622)

# ВВЕДЕНИЕ

Цели и задачи работы: изучение основных команд Linux ОС и порядка сборки проекта на языке С++ и Go.

# Ход выполнения

**1. Процессы, память**

**Команда top**

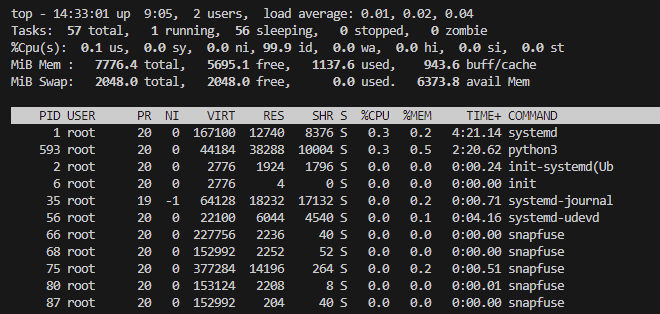
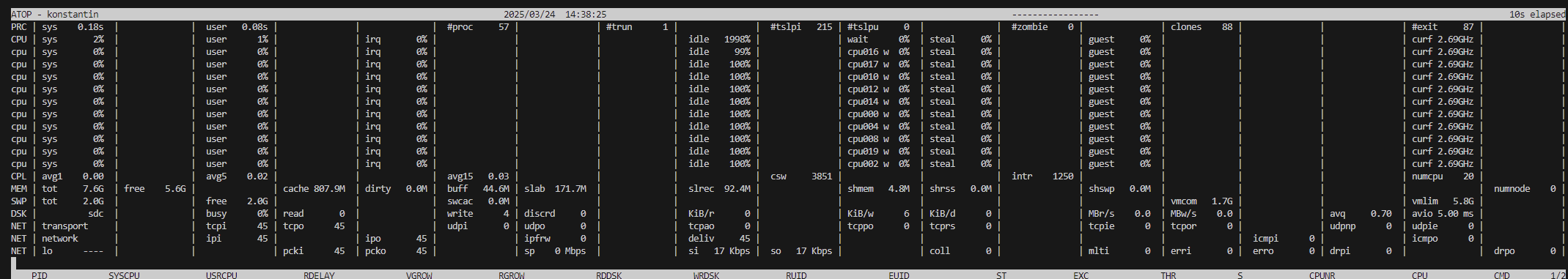
****

Рисунок 1 – вывод команды top

1. **Что такое «процесс» в Linux? Какие у него есть параметры?**  
   Процесс в Linux — это экземпляр выполняемой программы. У каждого процесса есть уникальный идентификатор (PID), состояние (например, running, sleeping, zombie), приоритет, использование памяти и CPU, и другие параметры.
2. **Чем отличаются типы памяти VIRT от RES в выводе top?**
   * **VIRT** — это общий объем виртуальной памяти, которую процесс использует (включая swap, shared memory и т.д.).
   * **RES** — это объем физической памяти, которую процесс использует в данный момент.
3. **Zombie-процессы – что это такое? Как они возникают? К чему приводит рост их количества в системе? Как с ними бороться?**  
   Zombie-процессы — это завершенные процессы, которые остаются в таблице процессов до тех пор, пока родительский процесс не прочитает их статус. Они возникают, когда родительский процесс не обрабатывает сигнал завершения дочернего процесса. Рост их количества может привести к заполнению таблицы процессов, что может вызвать проблемы с созданием новых процессов. Бороться с ними можно, завершая родительский процесс или перезапуская систему.
4. **Load Average – что по сути означает эта метрика? Какие ее значения можно считать приемлемыми, а какие указывают на перегрузку? Зависят ли нормальные метрики load average от типа CPU?**  
   Load Average показывает среднюю нагрузку на систему за последние 1, 5 и 15 минут. Значение меньше количества ядер CPU считается нормальным. Значения выше указывают на перегрузку. Нормальные метрики зависят от количества ядер CPU.
5. **%wa – о чем нам говорит данный параметр? К каким выводам можно прийти, если он выходит из нормы?**  
   %wa показывает процент времени, которое CPU проводит в ожидании завершения операций ввода-вывода (I/O). Если этот параметр высокий, это может указывать на проблемы с диском или сетью.

**Команда atop**

****

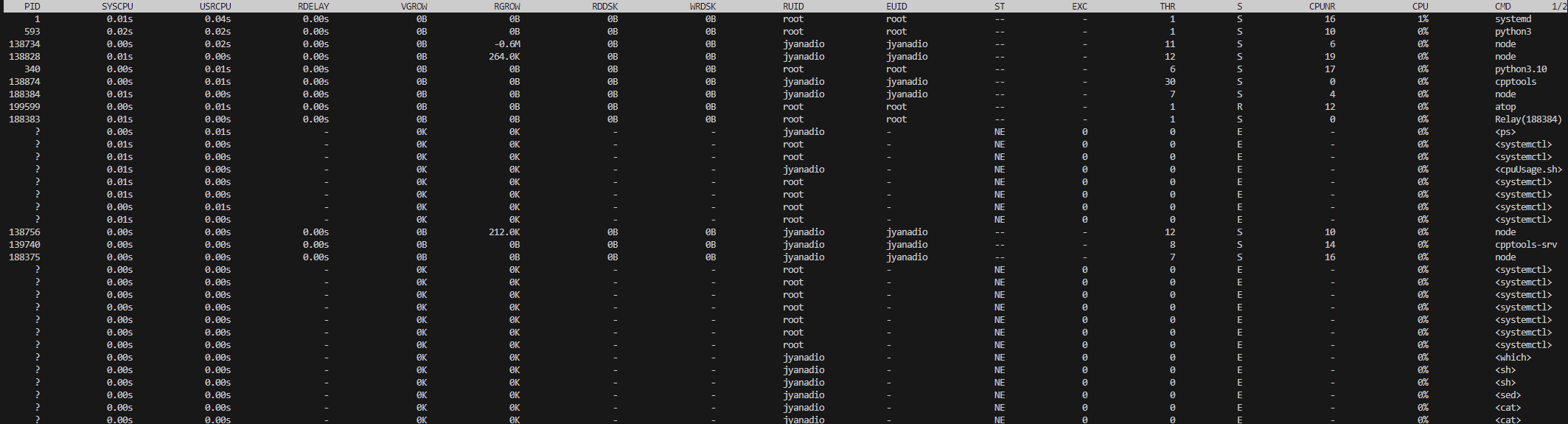
****

Рисунок 2 – вывод команды atop

1. **Что есть такого в atop, чего нет в top? Какие дополнительные сведения о системе он дает?**  
   atop предоставляет более детальную информацию о процессах, включая использование диска, сети, и историю изменений. Он также показывает нагрузку на систему за определенный период времени.

**Команда iotop**

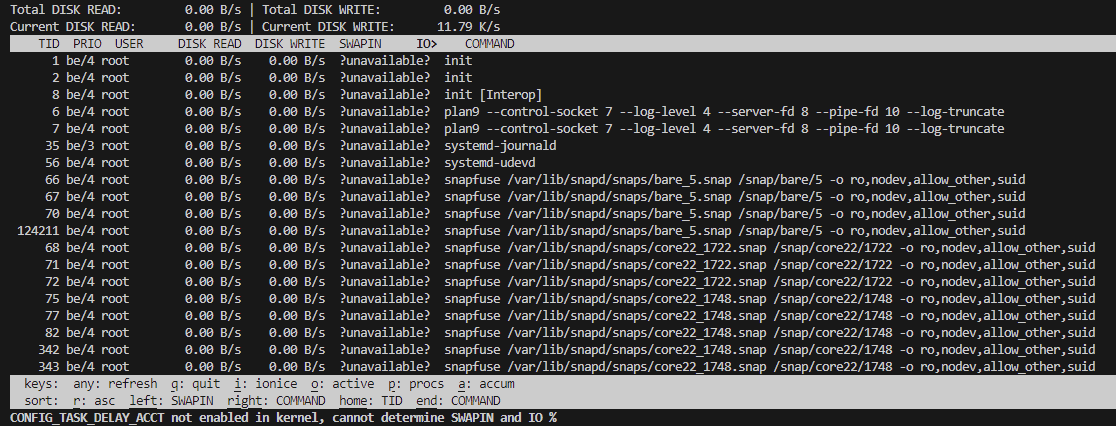


Рисунок 3 – вывод команды iotop

1. **На чем специализируется данная утилита? Почему нам не хватает только top? Какую проблему она позволит нам решить?**  
   iotop специализируется на мониторинге операций ввода-вывода (I/O) для каждого процесса. Она помогает выявить процессы, которые активно используют диск, что может быть полезно для диагностики проблем с производительностью.

**Команда lsof**

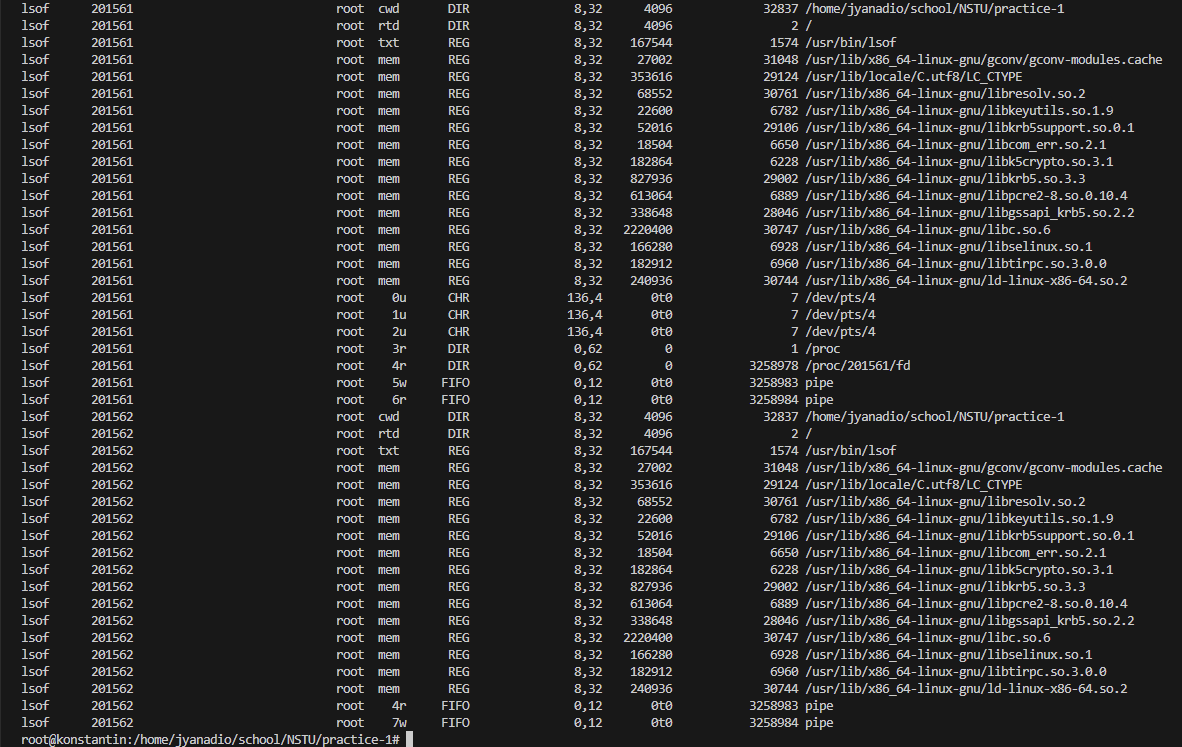


Рисунок 4 - Вывод команды lsof

1. **Зачем нам нужна информация о количестве открытых файлов и сокетов в системе? Какие есть ограничения в Linux по количеству открытых файлов и сокетов?**  
   Информация о количестве открытых файлов и сокетов помогает диагностировать утечки ресурсов и проблемы с производительностью. В Linux есть ограничения на количество открытых файлов и сокетов, которые можно настроить через параметры ядра (например, ulimit).

**Команда lscpu**

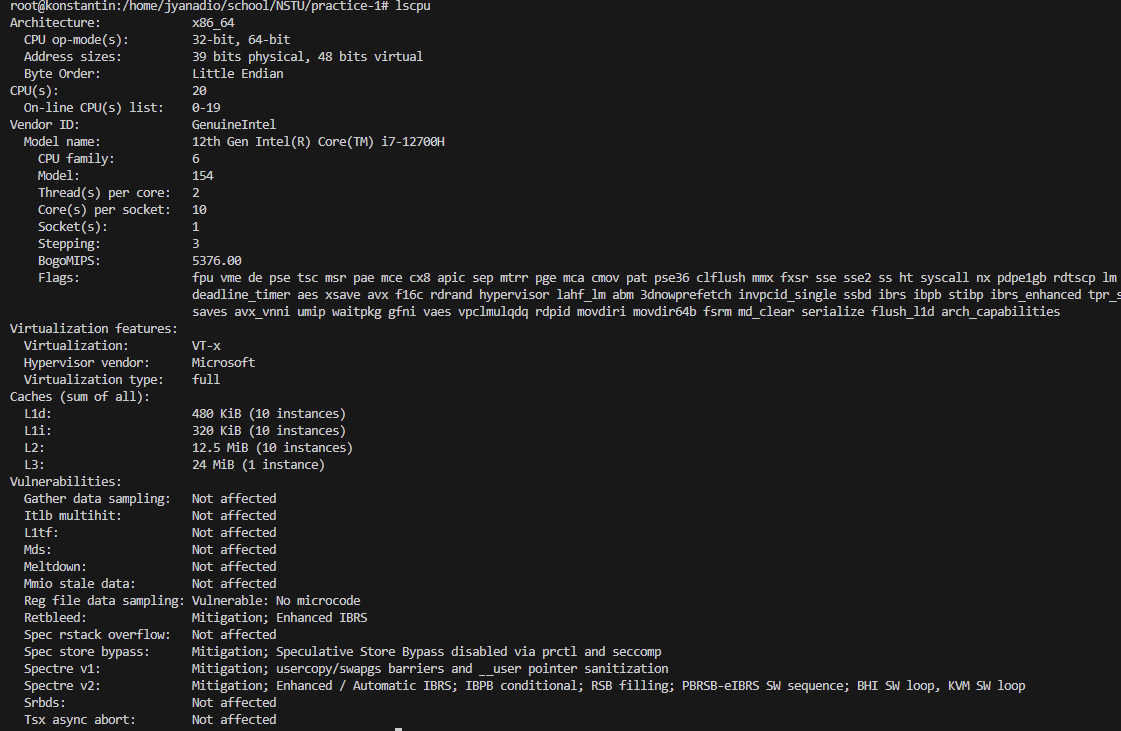


Рисунок 5 – вывод команды lscpu

1. **В выводе команды обратить внимание на Flags. Что это за флаги?**  
   Флаги в выводе lscpu показывают поддерживаемые функции процессора, такие как виртуализация (VT-x), поддержка 64-битных инструкций (lm), и другие.

**Команда kill**

1. **Чем отличается выполнение команд kill от kill -9? В каких случаях нам может помочь ключ «-9»?**  
   kill отправляет сигнал завершения процессу, который может быть обработан или проигнорирован. kill -9 отправляет сигнал SIGKILL, который принудительно завершает процесс без возможности обработки. Ключ «-9» полезен, когда процесс не реагирует на обычные сигналы завершения.

**Команда free**

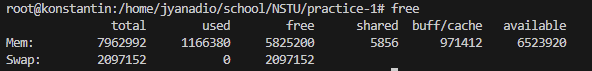


Рисунок 6 – вывод команды free

1. **Чем отличается свободная (free) память от доступной (available)? Из каких типов памяти складывается общая память?**
   * **Свободная (free) память** — это память, которая не используется.
   * **Доступная (available) память** — это память, которая может быть использована без необходимости освобождать кэш или буферы.  
     Общая память складывается из занятой (used) и свободной (free) памяти.

**Команда swapon/swapoff**



Рисунок 7 – вывод команды swapon

1. **Зачем в системе в принципе выключать swap? А зачем включать? Какие у swap плюсы и минусы?**
   * **Выключение swap** может быть полезно для тестирования или на системах с большим объемом оперативной памяти.
   * **Включение swap** позволяет системе использовать диск как дополнительную память, что полезно при нехватке оперативной памяти.
   * **Плюсы swap**: предотвращает сбои при нехватке памяти.
   * **Минусы swap**: использование диска медленнее, чем оперативной памяти.

**2. Утилиты управления**

**Команда apt**

Это менеджер пакетов командной строки, предоставляющий команды для поиска и управления, а также для запроса информации о пакетах. Он обеспечивает ту же функциональность, что и специализированные инструменты APT, такие как apt-get и apt-cache, но по умолчанию включает опции, более подходящие для интерактивного использования.

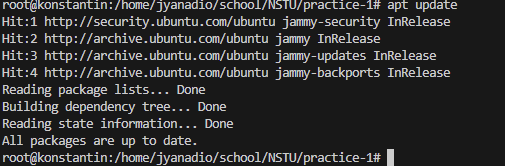


Рисунок 8 – вывод команды apt update

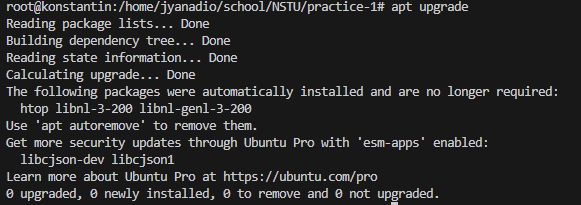


Рисунок 9 – вывод команды apt upgrade

1. **Чем отличается apt update от apt upgrade?**
   * apt update обновляет список доступных пакетов из репозиториев.
   * apt upgrade обновляет установленные пакеты до последних версий.
2. **Откуда команда apt знает, в каких репозиториях нужно искать пакеты? Где хранится информация о репозиториях?**  
   Информация о репозиториях хранится в файле /etc/apt/sources.list.
3. **Как добавить новый репозиторий?**  
   Новый репозиторий можно добавить, отредактировав файл /etc/apt/sources.list или создав файл в директории /etc/apt/sources.list.d/.

**Команда dpkg**

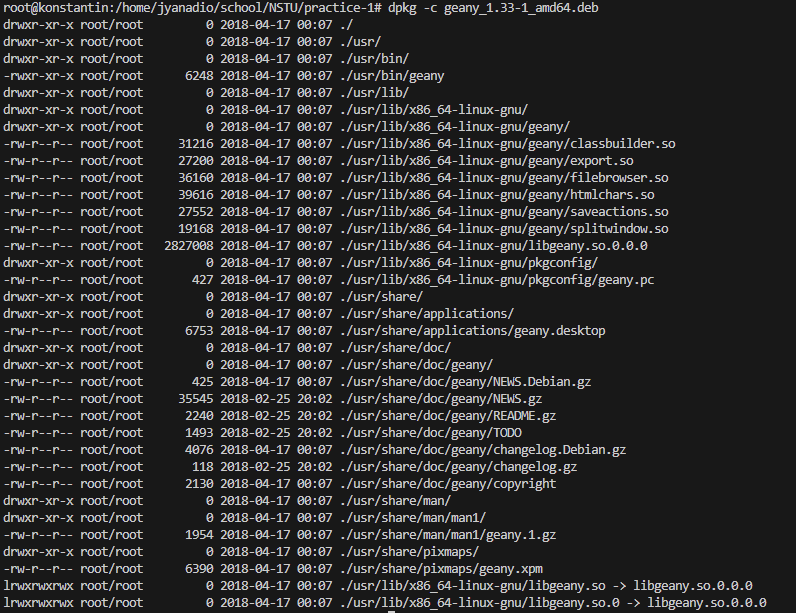


Рисунок 10 – просмотр содержимого deb пакета с помощью dpkg

1. **В чем разница между установкой пакета с помощью dpkg и apt?**  
   dpkg устанавливает пакет без разрешения зависимостей, тогда как apt автоматически разрешает и устанавливает зависимости.
2. **Как посмотреть, что внутри deb пакета?**  
   Используйте команду dpkg -c <имя\_пакета.deb>.
3. **В чем разница между ключами -r и -P?**
   * -r удаляет пакет, но оставляет конфигурационные файлы.
   * -P полностью удаляет пакет и его конфигурационные файлы.

**Команда crontab**

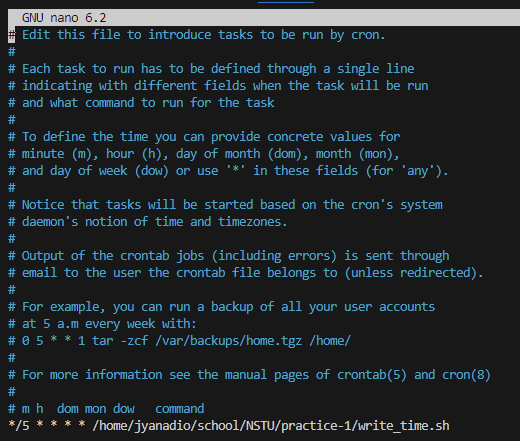


Рисунок 11 – добавление задачи в crontab (запуск скрипта вывода времени в файл каждые 5 минут)

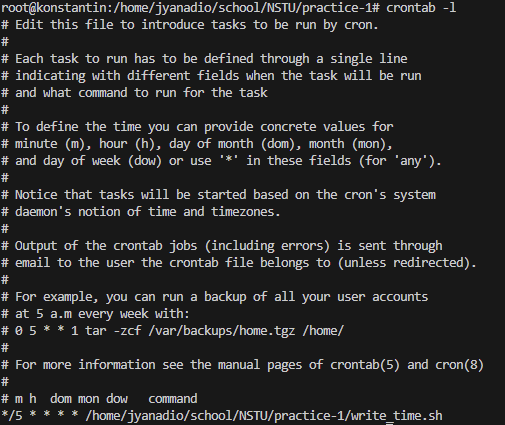


Рисунок 12 – просмотр текущих задач в кронтаб

1. **Какой будет текущая директория при запуске скрипта в CRON по относительному пути?**  
   Текущая директория будет домашней директорией пользователя, от имени которого выполняется задача.
2. **Что будет, если создать задачу в CRON по запуску скрипта с владельцем root из под УЗ обычного пользователя?**  
   Задача будет выполняться с правами root, но создана она будет от имени обычного пользователя.

**Команда systemctl**

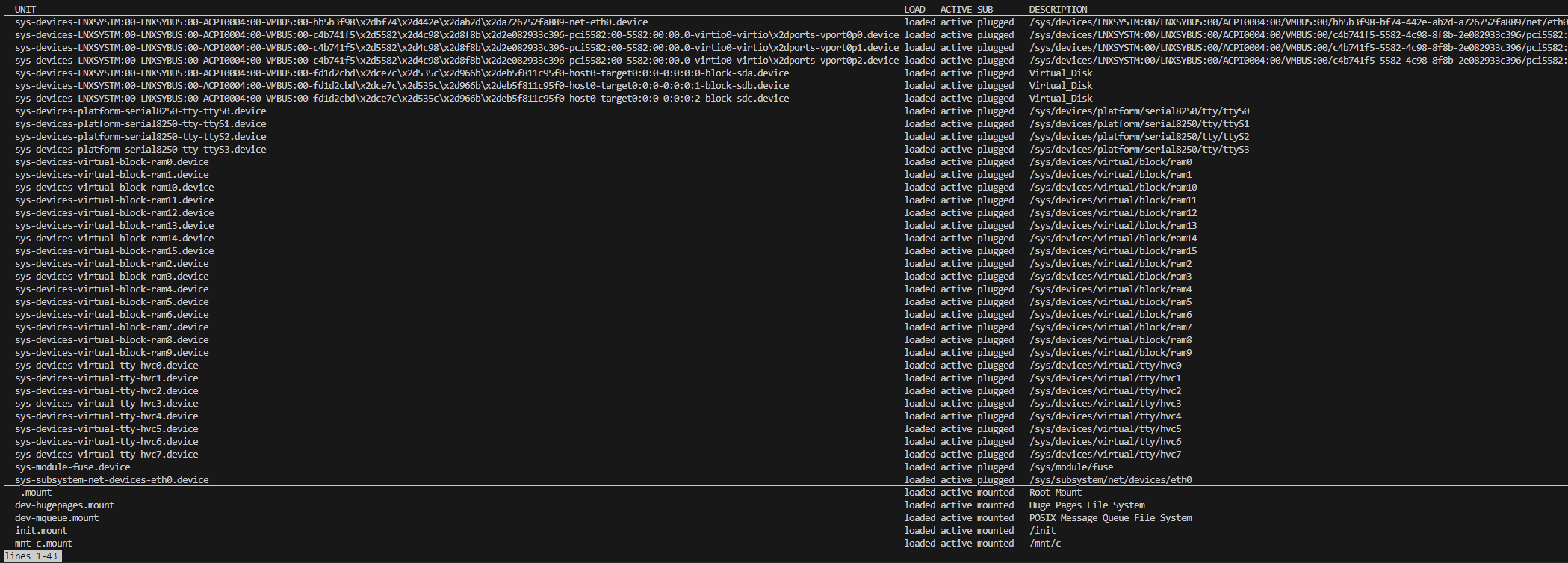


Рисунок 13 – вывод команды systemctl

1. **В каких директориях и файлах в Linux конфигурируется Systemd?**  
   Systemd конфигурируется в директориях /etc/systemd/system/ и /lib/systemd/system/.

**Команда man**

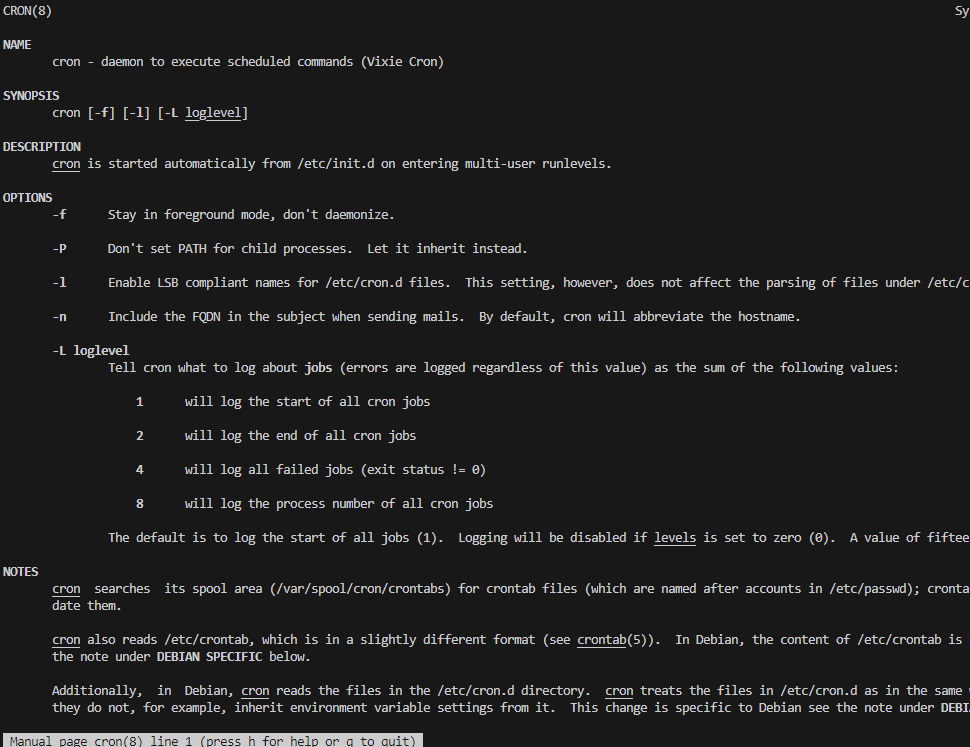


Рисунок 14 - вывод команды man cron

1. **Что происходит при запуске команды man?**  
   Команда man отображает руководство по использованию другой команды.

**Команда env**



Рисунок 15 – вывод команды env

1. **Что показывает вывод команды env?**  
   Команда env показывает переменные окружения текущей сессии.

**3. Файлы, пользователи**

**Команда ln**

1. **Зачем мы можем использовать символические ссылки?**  
   Символические ссылки позволяют создавать ссылки на файлы или директории, что упрощает доступ к ним из разных мест.
2. **Что будет, если создать символическую ссылку на файл по относительному пути, а потом переместить эту ссылку в другую директорию?**  
   Ссылка может перестать работать, если относительный путь изменится.

**Команда tar**

1. **Как заархивировать все файлы в директории в .tar.gz архив?**

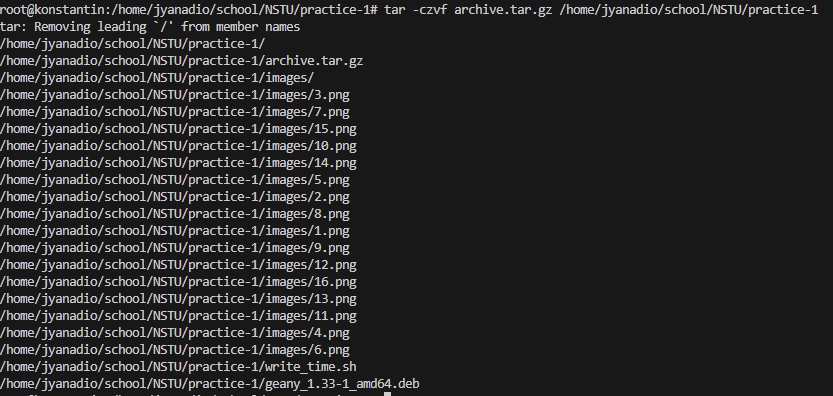


Рисунок 16 – архивация файлов в директории в .tar.gz архив

1. **Как разархивировать .tar.gz архив в определенную директорию?**

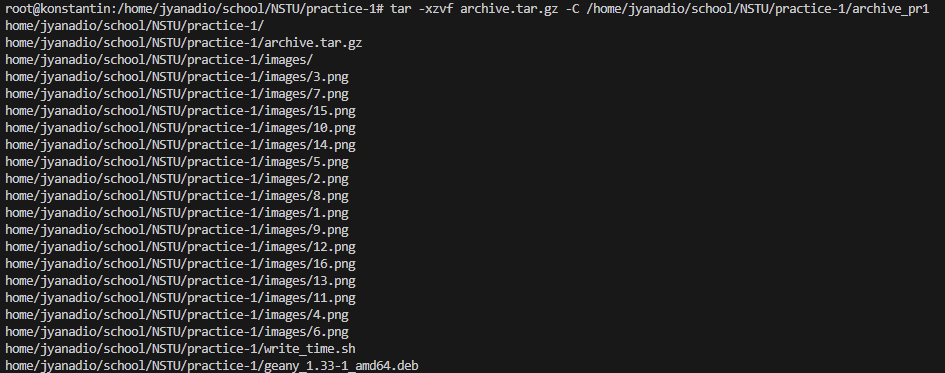
****

Рисунок 17 – разархивация .tar.gz архива

1. **Как создать просто .tar архив?**

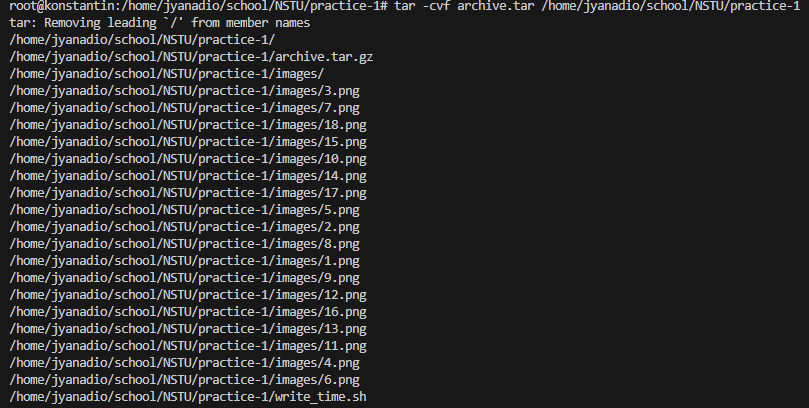
****

Рисунок 18 – создание .tar архива

1. **Что будет, если попробовать разархивировать .tar.gz архив размером 10Gb?**  
   Архив будет разархивирован, если на диске достаточно места.

**Команда adduser**

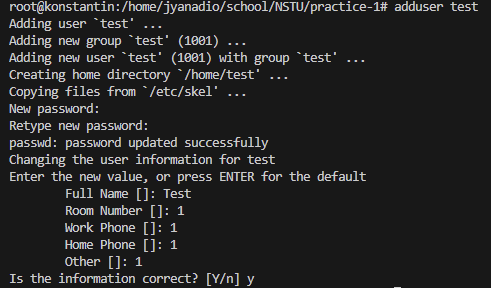


Рисунок 19 – создание нового юзера

1. **Как еще можно создать пользователя в системе?**  
   Можно использовать команду useradd.

**Команда chmod**

1. **Что будет, если сделать chmod 777 filename?**  
   Файл станет доступен для чтения, записи и выполнения всем пользователям.
2. **Что значат три семерки?**
   * Первая 7: права для владельца (rwx).
   * Вторая 7: права для группы (rwx).
   * Третья 7: права для остальных (rwx).
3. **Что будет, если сделать chmod g-x filename?**  
   У группы будет отозвано право на выполнение файла.

**Команда chown**

1. **Что значит chown nobody:nogroup filename?**  
   Это изменит владельца файла на пользователя nobody и группу nogroup.

**Команда sudo**

1. **Как можно сделать пользователя привилегированным?**  
   Добавить пользователя в группу sudo или отредактировать файл /etc/sudoers.

**Команда su**

1. **Чем отличается su от sudo?**  
   su переключает пользователя на root, тогда как sudo выполняет команду с правами root.

**4. Текст**

**Команда awk**

1. **Задача: вывести на экран ip-адрес хоста, обработав вывод команды ifconfig.**

****

Рисунок 20 – результат работы ifconfig

**Команда wc**

1. **Что покажет команда lsof | wc?**  
   Она покажет количество строк, слов и символов в выводе lsof.

Рисунок 21 – вывод команды lsof | wc

**Команда grep**

1. **Что делают ключи -C, -B, -A, -v?**
   * -C: показывает строки вокруг совпадения.
   * -B: показывает строки перед совпадением.
   * -A: показывает строки после совпадения.
   * -v: инвертирует поиск, показывая строки, которые не совпадают.

**Команда cat**

1. **Что будет, если запустить команду без аргументов?**  
   Команда cat будет ожидать ввода с клавиатуры.

**Команда date**

1. **Задача: вывести на экран текущую дату в формате YYYY-MM-DD.**



Рисунок 22 – вывод текущей даты и времени

**Команда vi/nano**

1. **Задача: с помощью редактора vi создать текстовый файл со следующим содержимым - "Linux is so fun!".**

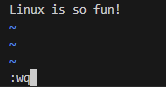
****

Рисунок 23 – создание текстового файла в vim

**Команда echo**

1. **Задача: вывести текущего пользователя, используя переменные окружения.**

****

Рисунок 24 – вывод текущего пользователя используя переменные окружения

**Команда less**

1. **Когда полезно применение команды less?**  
   less полезен для просмотра больших файлов постранично.

**Команда sed**

1. **Задача: вывести текущий ip-адрес без маски сети.**

****

Рисунок 25 – вывод текущего ip без маски

**5. Файлы**

**Файл /etc/passwd**

1. **Что значит строка /usr/sbin/nologin в содержимом файла?**  
   Это означает, что пользователь не может войти в систему через shell.

**Файл /etc/shadow**

1. **Чем отличается от /etc/passwd?**  
   /etc/shadow содержит зашифрованные пароли пользователей, тогда как /etc/passwd содержит информацию о пользователях.

**Файл /etc/sudoers**

1. **Как можно редактировать файл безопасно?**  
   Используйте команду visudo, которая проверяет синтаксис перед сохранением.

**Файл /etc/resolv.conf**

1. **Как добавить новый DNS-сервер?**  
   Добавьте строку nameserver <IP-адрес> в файл /etc/resolv.conf.

**Файл /var/log/syslog**

1. **Что показывает файл?**  
   Файл /var/log/syslog содержит системные логи.
2. **Как туда что-нибудь записать?**  
   Используйте команду logger для записи в системный лог.

**6. Процесс сборки программы на C++**

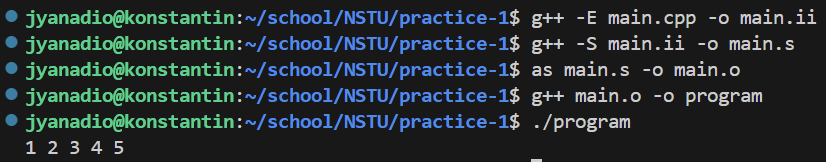


Рисунок 26 – команды выполненные в процессе сборки

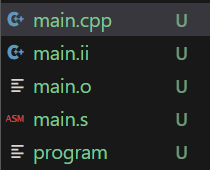


Рисунок 27 - файлы полученные в процессе сборки

**Препроцессинг**

*g++ -E main.cpp -o main.ii*

**Компиляция**

*g++ -S main.ii -o main.s*

**Ассемблирование**

*as main.s -o main.o*

**Компоновка**

*g++ main.o -o program*

**Запуск программы**

*./program*

**7.** **Процесс сборки программы на Go.**

**С помощью команды***go run <имя\_файла>.go***.**Она автоматически компилирует исполняемый файл, запускает его и сразу же удаляет. Этот способ используется в тех случаях, когда нужно разово запустить небольшой кусок кода и быстро получить результат.



Рисунок 28 – сборка и запуск программы на go с помощью run

**С помощью команды***go build <имя\_файла>.go***.** Она также занимается компиляцией и создает в директории исполняемый файл, способный запускаться на разных ОС. Особенностью этой команды является то, что она обеспечивает кроссплатформенность, так как позволяет контролировать, для какой ОС и архитектуры создается бинарный файл.

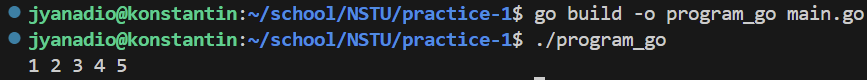


Рисунок 29 – сборка и запуск программы на go с помощью build

**Что происходит во время выполнения** *go build.* Дляпросмотра необходимо использовать флаг -x.

**Go создает временный файл конфигурации импорта, который используется для управления зависимостями во время компиляции.**

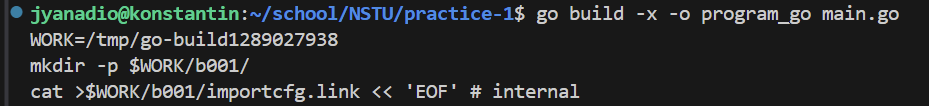


Рисунок 30 – процесс сборки проекта на go

**Пакеты и библиотеки были скомпилированы и будут использованы в процессе сборки вашего приложения на Go.**

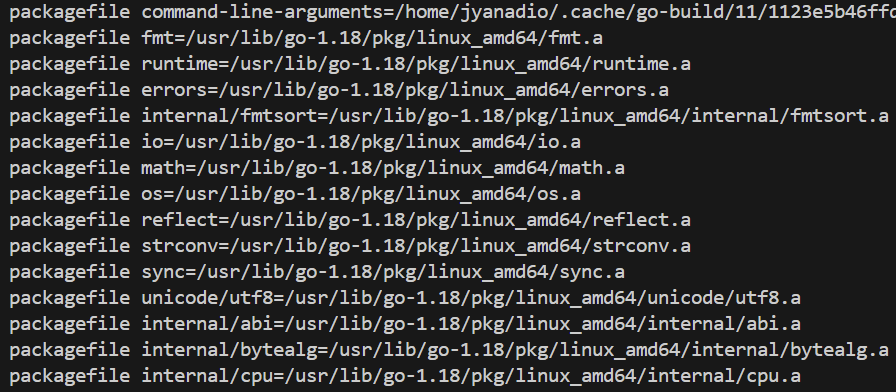


Рисунок 31 – процесс сборки программы на go

**Go выполняет линковку и создает исполняемый файл, а также управляет временными файлами и метаданными сборки.**

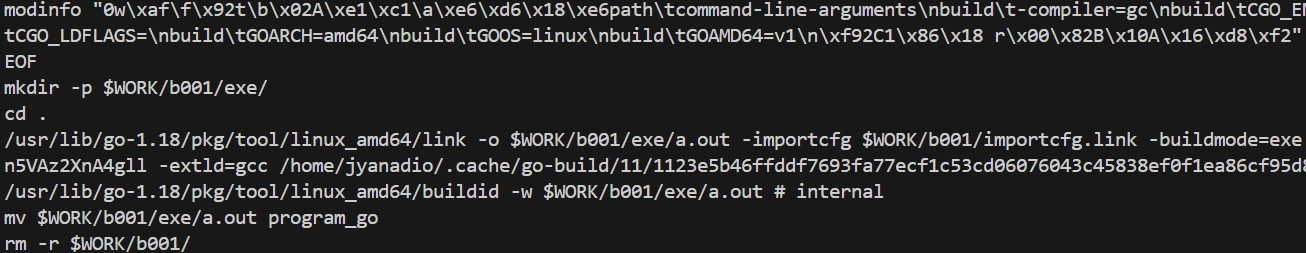


Рисунок 32 – процесс сборки программы на go

**8. Makefile с комментариями по процессу сборки**

Листинг кода Makefile

all: program\_c program\_go

program\_go: main.go

    @echo "Сборка проекта go"

    go build -o program\_go main.go

program\_c: preprocess\_c compilation\_c assembly\_c main.o

    @echo "Компоновка"

    g++ main.o -o program

preprocess\_c: main.cpp

    @echo "Препроцессинг"

    g++ -E main.cpp -o main.ii

compilation\_c: main.ii

    @echo "Компиляция"

    g++ -S main.ii -o main.s

assembly\_c: main.s

    @echo "Ассемблирование"

    as main.s -o main.o

clean:

    rm -rf \*.o \*.ii \*.s program\_go program

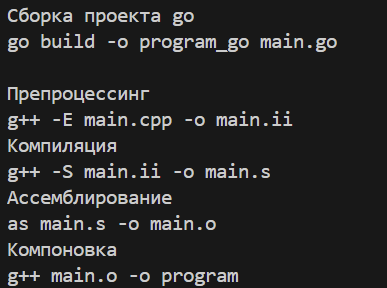


Рисунок 33 – результат работы Makefile

# Вывод

В результате данной лабораторной работы по программированию были выполнены все поставленные цели и задачи, нацеленные на изучение порядка сборки проекта на языке C++ и Go через консольную командную строку в Linux.