**Санкт-Петербургский политехнический университет Петра Великого**
**Институт компьютерных наук и кибербезопасности**
**Высшая школа программной инженерии**

# Лабораторная работа №6

# «Конфигурация и установка ядра Linux»

**Выполнила: Севостьянова Анна Викторовна**

**Группа: 5130904/30002**

**Преподаватель: Петров Александр Владимирович**

**Санкт-Петербург**

**2024г.**

**Оглавление**

1. Введение ..... 3

1.1 Актуальность ..... 3

1.2 Цель ..... 4

1.3 Задачи ..... 4

2. Основная часть ..... 5

2.1 Методы решения задач ..... 5

2.2 Основные результаты ..... 9

3. Заключение ..... 10

3.1 Выводы по результатам проделанной работы ..... 10

4. Список использованных источников ..... 12

# 1. Введение

Данная лабораторная работа по сборке ядра Linux нацелена на ознакомление с процессом сборки и настройки ядра операционной системы.

Ядро Linux является одним из ключевых компонентов операционной системы, ответственным за управление ресурсами компьютера, обеспечение связи между аппаратным обеспечением и программным обеспечением, а также обеспечение безопасности и стабильности работы системы.

## 1.1 Актуальность

Сборка ядра Linux может быть актуальной по нескольким причинам, в зависимости от потребностей и целей пользователя или организации, например:

*1. Оптимизация и настройка.*

Настройка ядра Linux для определенных нужд. При сборке собственного ядра можно избирательно включать или исключать определенные функции, модули и драйверы, что позволяет оптимизировать работу системы под конкретные цели.

*2. Поддержка нового оборудования.*

Иногда актуальные ядра Linux содержат драйверы для нового аппаратного обеспечения, которое еще не было включено в стабильные версии дистрибутивов. Можно избавиться от данной проблемы, собрав более новое ядро.

*3. Изучение и обучение.*

Для студентов, исследователей и технических специалистов сборка ядра Linux является отличным способом изучения архитектуры операционной системы и углубления знаний в области компиляции программного обеспечения.

*4. Безопасность и настройка безопасности.*

Пользователи могут хотеть настроить ядро Linux для повышения безопасности своей системы путем включения или исключения определенных функций ядра.

*5. Производительность.*

Сборка ядра Linux с определенными параметрами может повысить производительность системы в целом, учитывая особенности конкретного аппаратного обеспечения.

Важно отметить, что большинству пользователей обычно не требуется собирать собственное ядро Linux, так как стандартные ядра, поставляемые с дистрибутивами, обеспечивают более чем достаточную функциональность для большинства задач.

### 1.2 Цель

Целью данной лабораторной работы является получение сконфигурированного и собранного ядра Linux.

### 1.3 Задачи

1. Подготовка системы
2. Установка исходных кодов ядра
3. Конфигурация ядра
4. Сборка ядра
5. Установка ядра

## 2. Основная часть

Информация о системе, на которой проводилось выполнение лабораторной работы:

| Linux | Memory | Cpu |
|---|---|---|
| Debian 12 | 15 Gi (system memory)<br>128 KiB L1dcache<br>128 KiB L1i cache<br>1 MiB L2 cache<br>6 MiB L3 cache | Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz<br>bus info: cpu@0<br>physical id: 4<br>version: 6.142.12<br>size: 4034MHz<br>capacity: 4200MHz<br>width: 64 bits |

### 2.1 Методы решения задач

*1) Первый пункт выполнения работы, и первая задача — это подготовка системы.*

sudo apt install build-essential libncurses-dev bison flex libssl-dev libelf-dev

вывод:

Чтение списков пакетов… Готово

Построение дерева зависимостей… Готово

Чтение информации о состоянии… Готово

Будут установлены следующие дополнительные пакеты:

binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-11 gcc gcc-11 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libfakeroot libfile-fcntllock-perl libfl-dev libfl2 libgcc-11-dev libitm1

liblsan0 libnsl-dev libquadmath0 libsigsegv2 libstdc++-11-dev libtirpc-dev
...

2) Следующий этап — получение исходных кодов ядра.

«tar -xvf linux-6.8.2.tar.xz»

3) Следующий этап - конфигурация
cp /boot/config-$(uname -r) ~/Download/linux-6.8.2/.config

make oldconfig

(Заполнение всех новых конфигураций)

4) Следующий этап — сборка ядра

создание файла со скриптом и последующий запуск

скрипт внутри:

```
#!/bin/bash
# Файл для записи результатов
output_file="times.csv"
echo "Threads,Time" > $output_file

# Перечислите желаемое количество потоков
threads_list="1 2 3 4 5 6 7 8 9"

for threads in $threads_list; do
```

```
    echo "Сборка с $threads потоками..."
    # Запуск сборки и измерение времени
    start_time=$(date +%s)
    make -j$threads 2>&1 >/dev/null
    end_time=$(date +%s)

    # Расчет времени сборки
    build_time=$((end_time - start_time))
    echo "$threads,$build_time" >> $output_file
# Очистить после сборки для следующего запуска
    make clean
done

echo "Сборка завершена. Результаты в файле $output_file"
```

вывод на каждом этапе:

```
сборка с 1 потоками...
  CLEAN   arch/x86/boot/compressed
  CLEAN   arch/x86/boot
  CLEAN   arch/x86/entry/vdso
  CLEAN   arch/x86/kernel/cpu
  CLEAN   arch/x86/kernel
  CLEAN   arch/x86/realmode/rmGenerating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.76
  CLEAN   arch/x86/tools
  CLEAN   arch/x86/lib
  CLEAN   certs
  CLEAN   drivers/firmware/efi/libstub
  CLEAN   drivers/scsi
```

CLEAN drivers/tty/vt

CLEAN init

CLEAN lib

CLEAN net/wireless

CLEAN security/selinux

CLEAN usr

CLEAN .

CLEAN modules.builtin modules.builtin.modinfo .vmlinux.export.c

…

5) Следующий этап — установка ядра

a)установка нужных модулей sudo make modules_install

вывод:

INSTALL /lib/modules/6.8.2/kernel/arch/x86/crypto/aegis128-aesni.ko

…

b) копирование ядра в /boot с помощью команды sudo make install

вывод:

INSTALL /boot

run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.8.2 /boot/vmlinuz-6.8.2

…

c)обновление загрузчика sudo update-grub

вывод:

Generating grub configuration file ...

Found linux image: /boot/vmlinuz-6.8.2

…

## 2.2 Основные результаты

После завершения процесса сборки и установки ядра были получены следующие результаты, представленные в Таблице 1.

| ***Количество потоков*** | ***Время сборки (минуты)*** |
|---|---|
| 1 | 22,8 |
| 2 | 12,5 |
| 3 | 13,7 |
| 4 | 7,8 |
| 5 | 7,77 |
| 6 | 8,3 |
| 7 | 7,42 |
| 8 | 7,35 |
| 9 | 7,38 |

*Таблица 1 «Время сборки (в минутах) в зависимости от количества задействованных потоков»*

Для большей наглядности полученных результатов построим график, представленный на Рисунке 1.

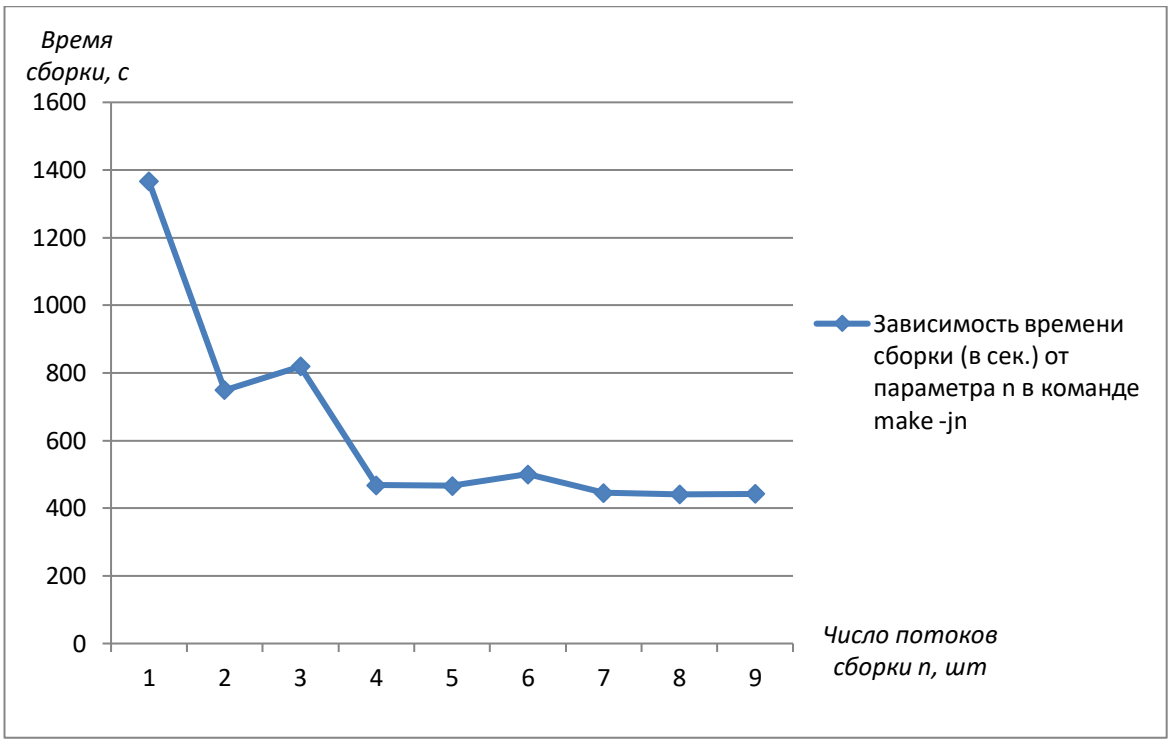


*Рисунок 1 «Зависимость времени сборки (в секундах) от количества задействованных потоков»*

## 3. Заключение

Данная лабораторная работы была посвящена сборке ядра Linux. Все выводы, сделанные по ее итогам, представлены в следующих пунктах.

### 3.1 Выводы по результатам проделанной работы

В результате работы:

1. Поставленная цель была достигнута. Собранное ядро успешно установилось и запустилось;

2. Поставленные задачи были решены.

3. Возникшие в ходе выполнения работы проблемы и трудности, которые были решены:

а) В соответствии с условием, требуется запустить программу с 0 потоков, однако это является невозможным.

б) Возникла ошибка из-за несовместимости конфигурации с ядром.

в) Перед запуском скрипта для проверки правильности выполнения задачи, была запущена сборка отдельно, что привело к кэшированию, а это повлекло некорректные результаты с одним потоком при запуске.

4. Итоги по необходимому числу потоков для эффективной сборки ядра на системе, на которой выполнялась работа:

Проанализировав таблицу (Таблица 1) и график (Рисунок 1), сформированные после цикла сборок ядра с разным числом потоков, был сделан вывод, что наибольшую эффективность имеет выполнение команды make –j8, т. е. с 8 потоками.

5. Сделанные выводы:

Сборка собственного ядра Linux позволяет оптимизировать работу системы под конкретные потребности, повысить производительность и эффективность обработки задач.

## 4. Список использованных источников

[1] https://www.kernel.org/