

Министерство образования и науки РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Лабораторная работа №6
“Сборка ядра Linux”

Работу выполнил студент
Кладковой Максим Дмитриевич
Группа: 5130904/30005
Руководитель: Петров Александр Владимирович

Содержание

| | |
|--|---|
| Содержание..... | 2 |
| Аппаратная платформа..... | 2 |
| Программная платформа..... | 2 |
| Задание..... | 3 |
| Подготовка к выполнению работы..... | 4 |
| Подготовка к сборке ядра..... | 4 |
| Сборка ядра:..... | 5 |
| Завершение и проверка на правильность установки:..... | 5 |
| Выполнение работы:..... | 5 |
| Заключение..... | 7 |
| Индивидуальное задание..... | 7 |
| Цель:..... | 7 |
| Задание:..... | 8 |
| Задачи:..... | 8 |
| Подготовка к выполнению работы:..... | 8 |
| Нахождение и патчей в исходном коде Debian и их применение к ванильному ядру:..... | 8 |
| Работа с ядром:..... | 8 |
| Вывод:..... | 9 |

Аппаратная платформа

Honor MagicBook HUAWEI NBLK-WAX9X

CPU: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx

Программная платформа

PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"

NAME="Debian GNU/Linux"

VERSION_ID="12"

VERSION="12 (bookworm)"

VERSION_CODENAME=bookworm

ID=debian

HOME_URL="https://www.debian.org/"

SUPPORT_URL="https://www.debian.org/support"

BUG_REPORT_URL="<https://bugs.debian.org/>"

Задание:

1. Установить исходный код ядра, предоставляемый вашим дистрибутивом (ванильная версия не рекомендуется).
2. Сконфигурировать и собрать ядро из установленных исходных файлов.
3. Протестировать систему с новым ядром.
4. Разработать сценарий, который запускает сборку ядра в цикле для $-jN$ со значениями от 1 до $2N+1$, где N – число ядер в системе, включая виртуальные.
5. Число ядер можно узнать по `cat /proc/cpuinfo`. Сценарий возвращает только время работы сборки на процессоре (используйте `time`, а все сообщения `make-kpkg` перенаправляйте в `/dev/null`). На каждой итерации очищайте дерево исходного кода (например, `make-kpkg clean`).
6. Предоставить отчет о проделанной работе. Дополнительно необходимо предоставить файл конфигурации ядра.
7. Отчет и файл конфигурации необходимо представить в виде архива, названного в соответствии со следующим шаблоном: <первая буква имени студента><фамилия студента><номер группы студента>.
8. После согласования с преподавателем предоставить отчет.

Цели:

1. Сконфигурированное и собранное ядро Linux.
2. Время сборки ядра при различном числе потоков сборки.
3. Нахождение оптимального числа потоков для сборки ядра.
4. Выполнение индивидуального задания.

Задачи:

1. Подготовка системы.
2. Установка исходного кода ядра.
3. Конфигурация ядра.
4. Сборка ядра.
5. Установка ядра.
6. Очищение дерева сборки.
7. Проверка работоспособности.
8. Написание сценария, собирающего ядро на потоках от 1 до $2N+1$ и выводящего время.
9. Поиск оптимального числа потоков для сборки ядра.

10. Собрать ядро для дистрибутива, используя исходный код, предоставляемый дистрибутивом.(для ИЗ).
11. Собрать ядро для дистрибутива, используя исходный код, предоставляемый ресурсом kernel.org, с наложением на него заплат от дистрибутива.(для ИЗ).
12. Подведение итогов.

Подготовка к выполнению работы

Подготовка к сборке ядра

1. Установка пакетов:

- fakeroot - Создает поддельную корневую среду.
- Xz-utils - Обеспечивает быстрое сжатие и распаковку файлов .
- Libssl-dev - Поддерживает SSL и TLS , которые шифруют данные и обеспечивают безопасность подключения к Интернету.
- Libelf-dev - Выпускает общую библиотеку для управления файлами ELF (исполняемые файлы, дампы ядра и объектный код).
- Bison - Преобразует описание грамматики в программу на языке C.
- Flex - Генератор лексических анализаторов.
- build-essential - Компиляция программного обеспечения.
- ncurses-dev - Библиотека для реализации текстового пользовательского интерфейса.
- bc - Интерактивный интерпретатор Си-подобного языка, позволяет выполнять вычисления с произвольно заданной точностью.
- Kernel-package - утилита для сборки пакетов Debian.

2. Загрузил и извлёк исходный код ядра:

- wget <https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.15.149.tar.xz> && tar xvf linux-5.15.149.tar.xz

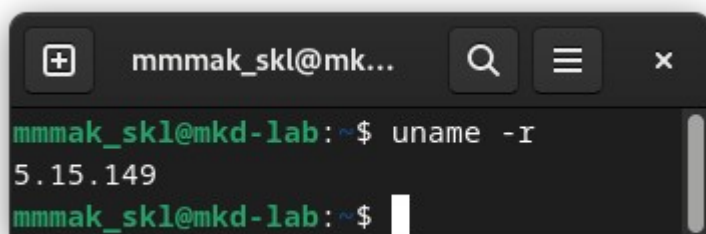
3. Перешел в каталог, который будет использоваться для сборки ядра и скопировал существующий файл конфигурации:

- cd Загрузки/linux-5.15.149
- cp -v /boot/config-\$(uname -r) .config

Сборка ядра:

1. `sudo make`
2. `sudo make modules`
3. `sudo make modules_install`
4. `sudo make install`
5. `reboot`

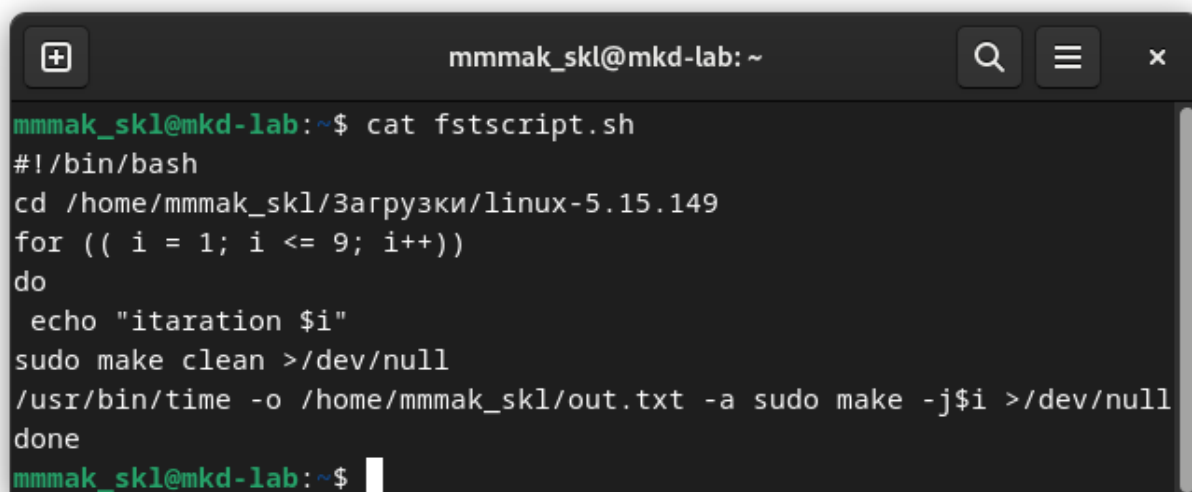
Завершение и проверка на правильность установки:



```
mmmak_skl@mkd-lab: ~$ uname -r
5.15.149
mmmak_skl@mkd-lab: ~$
```

Выполнение работы:

1. Написал скрипт для определения скорости сборки ядра на разном количестве потоков папо `fstscript.sh`



```
mmmak_skl@mkd-lab: ~$ cat fstscript.sh
#!/bin/bash
cd /home/mmmak_skl/Загрузки/linux-5.15.149
for (( i = 1; i <= 9; i++))
do
    echo "itaration $i"
    sudo make clean >/dev/null
    /usr/bin/time -o /home/mmmak_skl/out.txt -a sudo make -j$i >/dev/null
done
mmmak_skl@mkd-lab: ~$
```

Запуск скрипта:

1. `chmod +x fstscript.sh`

2. ./ ftscrip.sh

График соотношения количества потоков ко времени сборки:



N потоков

1
2
3
4
5
6
7
8
9

Время (min)

200:06
111:50
81:39
67:44
63:47
59:55
56:48
54:41
54:39

Заключение

1. В результате выполнения данной работы была достигнута цель поиска наиболее подходящего количества потоков при сборке ядра. Была выполнено 9 итераций сборки ядра.
2. Прделанные действия в ходе выполнения работы:
 - Установка зависимостей для сборки;
 - Скачивание исходного кода для ядра, поддерживаемого дистрибутивом;
 - Сборка ядра;
 - Установка собранного ядра;
 - Загрузка ядра;
 - Написание скрипта для определения скорости сборки ядра на разном количестве потоков;
 - Поиск оптимального количества потоков для сборки;
3. Самым оптимальным количеством потоков на аппаратной платформе является 8.

Индивидуальное задание

Цель:

Наложение заплат дистрибутива на ванильное ядро.

Задание:

Собрать ядро для вашего дистрибутива, используя исходный код, предоставляемый дистрибутивом, и используя исходный код, предоставляемый ресурсом kernel.org, с наложением на него заплат от вашего дистрибутива.

Задачи:

1. Установка необходимых пакетов для компиляции ядра
2. Загрузка исходного кода ядра с kernel.org
3. Загрузка исходного кода ядра Debian
4. Нахождение патчей в исходном коде Debian
5. Применение патчей к ванильному ядру
6. Настройка ядра
7. Компилирование ядра
8. Установка ядра
9. Обновление grub
10. Перезагрузка системы
11. Подведение итогов

Подготовка к выполнению работы

Загрузите исходный код ядра с [kernel.org](https://www.kernel.org):

```
wget https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.1.76.tar.xz
tar -xvf linux-6.1.76.tar.xz
cd linux-6.1.76/
```

Теперь загрузите исходный код ядра Debian:

```
apt-get source linux
```

Нахождение и патчей в исходном коде Debian и их применение к ванильному ядру

```
for i in debian/patches/*/*.patch; do patch -p1 < $i; done
```

Работа с ядром

Настройте ядро:

```
make menuconfig
```

Скомпилируйте и установите ядро:

```
make -j$(nproc)
sudo make modules
sudo make modules_install
sudo make install
```

Обновите grub:

```
sudo update-grub
```

Перезагрузите систему:

```
sudo reboot
```

Вывод

В результате выполнения данной работы была достигнута цель наложение заплат дистрибутива на ванильное ядро.