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
Мадьяров Глеб Сергеевич
Группа: 5130904/30002
Руководитель: Петров Александр Владимирович

Содержание

Оглавление

Аппаратная платформа

Micro-Star International Co., Ltd. Katana GF66 12UE
CPU: 2th Gen Intel® Core™ i5-12500H × 16

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 отчёт.

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. Используя git, скачать исходный код ядра, предоставляемый ресурсом kernel.org, наложить на него заплаты с использованием git, собрать ядро с заплатами
11. Подведение итогов.

Подготовка к выполнению работы

Подготовка к сборке ядра

1. Установка пакетов:
◦ Libssl-dev - Поддерживает SSL и TSL , 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 извлек исходный код ядра.
apt search ^linux-source
sudo apt install linux-source-6.1
mkdir ~/kernel
cd ~/kernel/
tar -xaf /usr/src/linux-source-6.1.tar.xz

3. Перешел в каталог, который будет использоваться для сборки ядра и скопировал существующий файл конфигурации:
cp /boot/config-6.1.0-18-amd64 ~/kernel/linux-source-6.1/.config
make olddefconfig

Сборка ядра:

sudo make -j16 deb-pkg
sudo dpkg -i ../linux-image-6.1.76_6.1.76-1_amd64.deb
sudo reboot
uname -r

Завершение и проверка на правильность установки:

```
gleb@Gleb:~$ uname -r
6.1.76
gleb@Gleb:~$
```

Выполнение работы:

1.Написал скрипт для определения скорости сборки ядра на разном количестве потоков nano Kernel_script.sh

```
gleb@Gleb:~$ cat Kernel_script.sh
#! /bin/bash
cd /home/gleb/kernel/linux-source-6.1
for ((i = 1; i <= 17; i++))
do
  echo "iteration $i"
sudo make clean >/dev/null
/usr/bin/time -o /home/gleb/out.txt -a sudo make -j$i deb-pkg >/dev/null
done
```

Запуск скрипта:
1. chmod +x Kernel_script.sh
2. ./Kernel_script.sh

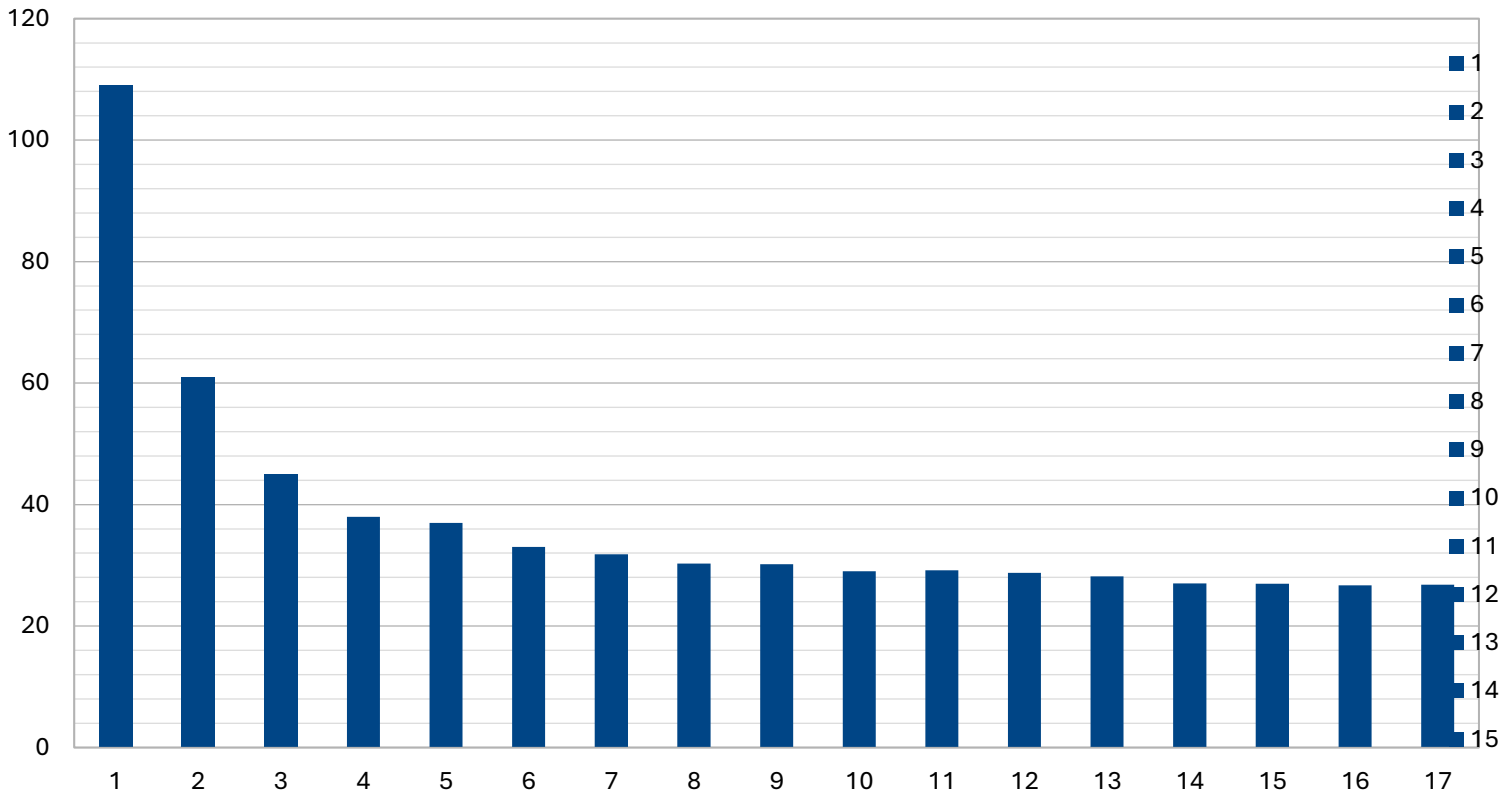

График соотношения количества потоков ко времени сборки:

N потоков	Время
1	1:49:19
2	1:01:41
3	44:56.07
4	38:15.27
5	37:10.77
6	33:20.73
7	31:45.33
8	30:24.62
9	30:17.83
10	29:03.71
11	29:17.08
12	28:45.13
13	28:13.41
14	27:03.11
15	26:56.76
16	26:38.59
17	26:47.80

Заключение

1. В результате выполнения данной работы была достигнута цель поиска наиболее подходящего количества потоков при сборке ядра. Было выполнено 17 итераций сборки ядра.
2. Проделанные 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 16.

Индивидуальное задание

Цель:

ядро с заплатами, собранное с использованием git.

Задание:

используя git, скачать исходный код ядра, предоставляемый ресурсом kernel.org, наложить на него заплаты с использованием git, собрать ядро с заплатами.

Задачи:

1. Загрузка исходного кода ядра используя git.
2. Нахождение патчей в исходном коде Debian
3. Применение патчей к ванильному ядру с использованием git.
4. Компилирование ядра
5. Установка ядра
6. Перезагрузка системы
7. Подведение итогов

Подготовка к выполнению работы

Загрузите исходный код ядра с использованием git:

sudo apt install -y git
git clone https://github.com/torvalds/linux.git
cd linux
git checkout v6.1

Теперь загрузите исходный код ядра Debian:

apt-get source linux

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

for P in $(ls ~/linux-6.1.90/debian/patches/**/*.patch); do git apply $P; 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 с использованием git.