

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

# Конфигурация и установка ядра Linux

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

Выполнил  
студент гр. в3530904/00030

В.С. Баганов

Руководитель  
ст. преподаватель, к.т.н.

С.А. Федоров

«\_\_\_\_\_» \_\_\_\_\_ 202\_\_ г.

Санкт-Петербург  
2023

# Содержание

<b>Постановка задачи</b>	<b>3</b>
<b>1. Подготовка системы к сборке и установке ядра</b>	<b>3</b>
<b>2. Конфигурация ядра для сборки</b>	<b>3</b>
2.1. Версия установленного ядра в системе . . . . .	3
2.2. Подготовка для компиляции ядра . . . . .	3
<b>3. Сборка ядра с различным числом потоков сборки</b>	<b>5</b>
<b>4. Результаты сборки с разными потоками и количеством ядер</b>	<b>6</b>
<b>5. Ускорение повторной сборки ядра с помощью ccache</b>	<b>7</b>
<b>6. Выводы</b>	<b>9</b>

# Постановка задачи

Конфигурация и установка ядра Linux

Ускорение повторной сборки ядра с помощью ccache

## 1. Подготовка системы к сборке и установке ядра

Все работы по конфигурации и установке ядра Linux производились на следующих платформах:

- Аппаратная платформа: Виртуальная машина Parallels Desktop 16 Версия 16.1.2 (49151) с конфигурацией 4 Гб ОЗУ, с 2-мя и 4-мя процессорами, на MacBook, 2,6 GHz, 2-ядерный процессор Intel Core i5, 8 Гб ОЗУ. ☒
- Программная платформа: Linux debian-gnu-linux-10-vm 4.19.0-16-amd64, ядро SMP Debian 4.19.181-1 (2021-03-19) x86\_64 GNU/Linux

Для сборки ядра были установлены пакеты :

- kernel-package
- fakeroot
- libncurses5-dev
- build-essential

Команда для установки пакетов

```
sudo apt-get install kernel-package fakeroot libncurses5-dev build-essential
```

## 2. Конфигурация ядра для сборки

### 2.1. Версия установленного ядра в системе

Чтобы понять версию ядра установленного на системе, вводим команду uname -a в результате получили

```
Linux debian-gnu-linux-10-vm 4.19.0-16-amd64
```

```
#1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
```

### 2.2. Подготовка для компиляции ядра

- Скачиваем версию нашего ядра  

```
sudo apt install kernel-source-4.19
```
- Создали директорию /kernel и переходим в нее  

```
mkdir /kernel; cd /kernel
```

- Распаковали ядро в директорию /kernel  
mkdir /kernel; cd /kernel

- Скопировали конфигурационный файл ядра нашей системы в директорию /kernel

```
cp /boot/config-4.19.0-5-amd64 /kernel/linux-source-4.19/.config
```

- Отключили подпись модулей и символьной информации ядра командами

```
scripts / config --disable MODULE_SIG
scripts / config --disable DEBUG_INFO
```

- Конфигурируем ядро, используя прежнюю конфигурацию

make olddefconfig конфигурирует ядро с настройками по умолчанию из старого ядра

- Делаем очистку make clean

- Собираем ядро make deb-pkg -j2

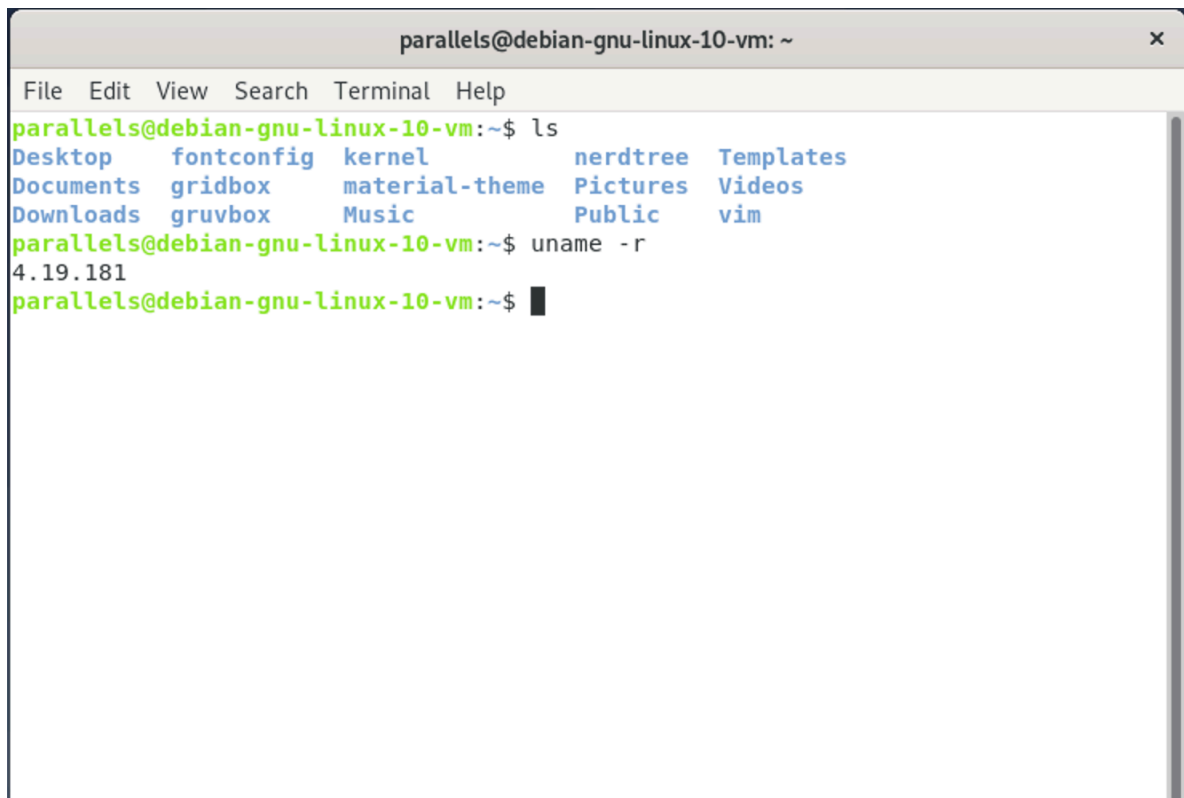
- Устанавливаем новое ядро dpkg -i linux-image-4.19.181\_4.19.181-1\_amd64.deb

- Проверяем установку ядра

```
reboot uname -r
```

— вывод первых 10 строк компиляции ядра

```
1  parallels@debian-gnu-linux-10-vm:~/kernel/linux-source-4.19$
   ↪ make deb-pkg -j2
2  make clean
3  /bin/bash ./scripts/package/mkdebian
4      TAR      linux-4.19.181.tar.gz
5  origversion=$(dpkg-parsechangelog -SVersion |sed
   ↪ 's/-[^-]*$//');\
6      mv linux-4.19.181.tar.gz
   ↪ ../linux-4.19.181_${origversion}.orig.tar.gz
7  dpkg-buildpackage -r"fakeroot -u" -a$(cat debian/arch) -i.git
   ↪ -us -uc
8  dpkg-buildpackage: info: source package linux-4.19.181
9  dpkg-buildpackage: info: source version 4.19.181-4
10 dpkg-buildpackage: info: source distribution buster
11 dpkg-buildpackage: info: source changed by parallels
   ↪ <parallels@debian-gnu-linux-10-vm.localdomain>
12 dpkg-buildpackage: info: host architecture amd64
```



```
parallels@debian-gnu-linux-10-vm: ~
File Edit View Search Terminal Help
parallels@debian-gnu-linux-10-vm:~$ ls
Desktop  fontconfig  kernel      nerdtree  Templates
Documents  gridbox    material-theme  Pictures  Videos
Downloads gruvbox    Music       Public    vim
parallels@debian-gnu-linux-10-vm:~$ uname -r
4.19.181
parallels@debian-gnu-linux-10-vm:~$
```

Рисунок 2.1. ядро успешно установлено

### 3. Сборка ядра с различным числом потоков сборки

С помощью скрипта, запускаем сборку ядра с количеством потоков от 1 до 16. Для более понятных результатов, делаем сборку на 2 и на 4 ядрах системы. Сборки проводились на старой версии ядра для чистоты результатов. Ниже код скрипта.

```
1 1 #!/bin/bash
2 2 start=`date +%s`
3 3 cd /home/parallels/kernel/linux-source-4.19
4 4   for ((var = 1; var ≤ 16; var++)); do
5 5       time make deb-pkg -j$var -s 2>/dev/null
6 6       echo "sborka j$var"
7 7       make clean
8 8   done
9 9 end=`date +%s`
10 10 runtime=$((end-start))
11 11 echo "Runtime was $runtime"
```

#### 4. Результаты сборки с разными потоками и количеством ядер

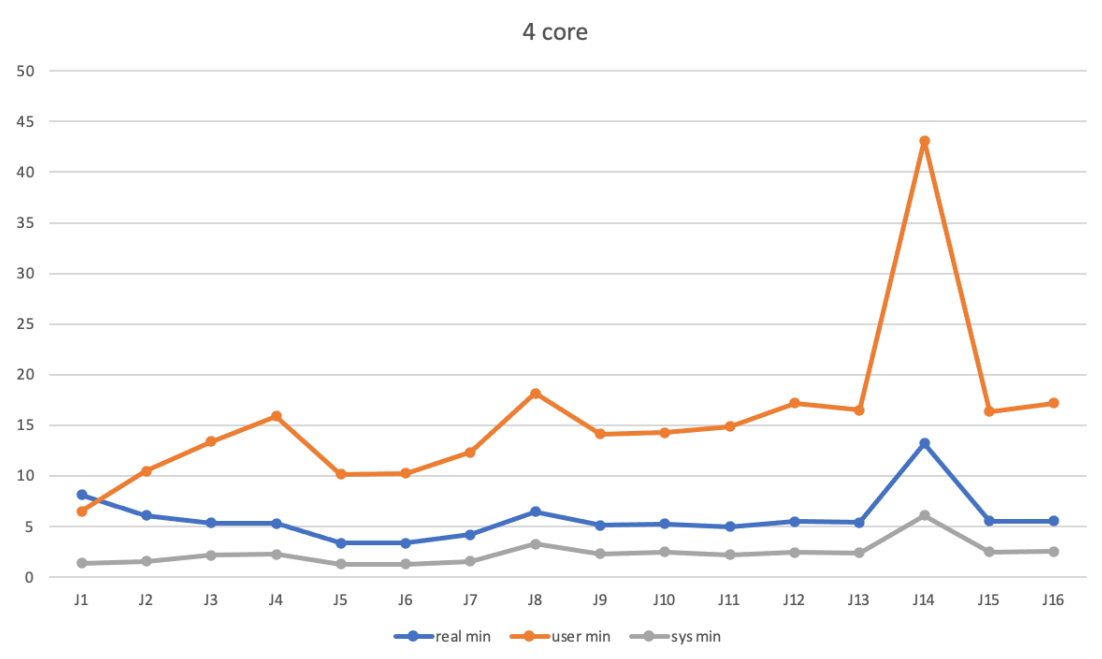


Рисунок 4.1. сборка на 4 ядрах

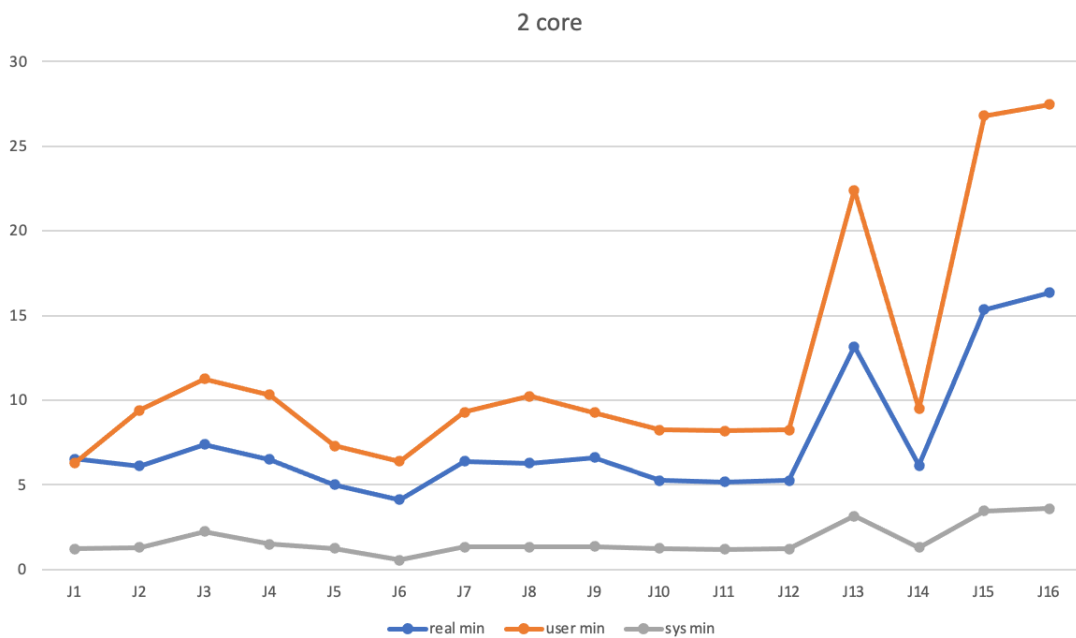


Рисунок 4.2. сборка на 2 ядрах

## 5. Ускорение повторной сборки ядра с помощью ccache

Установка ccache:

```
sudo apt install -y ccache  
sudo /usr/sbin/update-ccache-symlinks  
echo 'export PATH="/usr/lib/ccache:PATH"|tee -a ~/.bashrc  
source ~/.bashrc  
ccache -F 0  
ccache -M 0
```

Статистика кеша:

```
ccache -s
```

Очистить кеш и сбросить статистику:

```
ccache -C -z
```



```
parallels@debian-gnu-linux-10-vm: ~/kernel  
File Edit View Search Terminal Help  
parallels@debian-gnu-linux-10-vm:~/kernel$ ccache -s  
cache directory                /home/parallels/.ccache  
primary config                 /home/parallels/.ccache/ccache.conf  
secondary config              (readonly) /etc/ccache.conf  
stats updated                 Fri Jun  4 22:56:44 2021  
stats zeroed                  Fri Jun  4 22:56:44 2021  
cache hit (direct)            0  
cache hit (preprocessed)      0  
cache miss                     0  
cache hit rate                 0.00 %  
cleanups performed            0  
files in cache                 0  
cache size                     0.0 kB  
parallels@debian-gnu-linux-10-vm:~/kernel$
```

Рисунок 5.1. обнуление ccache

```
parallels@debian-gnu-linux-10-vm: ~/kernel
File Edit View Search Terminal Help
real    2m12.739s
user    4m1.616s
sys     1m10.021s
sborka j16
Runtime was 2915
parallels@debian-gnu-linux-10-vm:~/kernel$ ^C
parallels@debian-gnu-linux-10-vm:~/kernel$ ccache -s
cache directory                /home/parallels/.ccache
primary config                  /home/parallels/.ccache/ccache.conf
secondary config (readonly)    /etc/ccache.conf
stats updated                   Sat Jun  5 00:27:45 2021
stats zeroed                   Fri Jun  4 23:38:30 2021
cache hit (direct)              10253
cache hit (preprocessed)        16
cache miss                      763
cache hit rate                  93.08 %
called for link                 304
called for preprocessing        4732
unsupported code directive      32
no input file                   9740
cleanups performed              0
files in cache                  2288
cache size                      271.2 MB
parallels@debian-gnu-linux-10-vm:~/kernel$
```

Рисунок 5.2. результат работы ccache

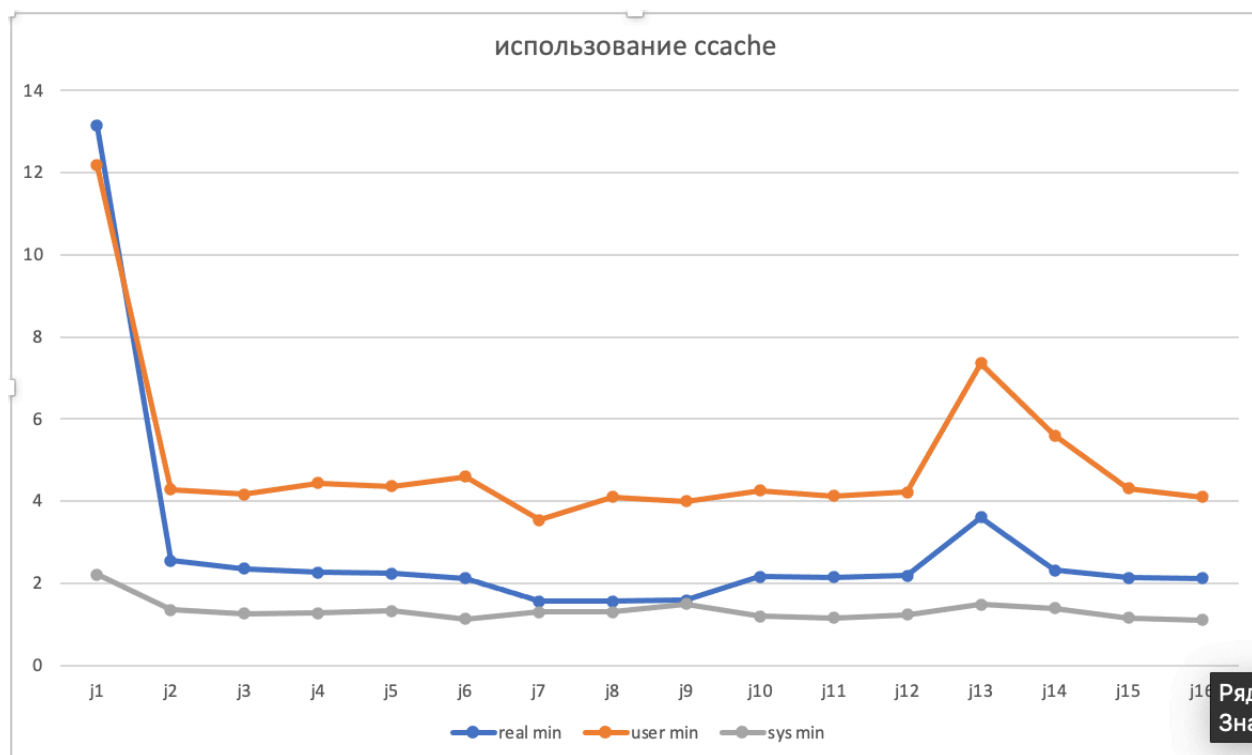


Рисунок 5.3. график времени работы ccache



## 6. Выводы

Из полученных данных можно сделать вывод, что минимальное время компиляции ядра было на j5 и j6. При чем на 2 и 4 ядрах данные схожи. Можно сделать вывод, что для данной программной и аппаратной системы для быстрой компиляции ядра подходит сборка в 6 потоков.

ссасhe позволяет экономить время компиляции файлов за счет кеширования файлов. Особенность ссасhe в том, что при первой компиляции затрачивается чуть больше времени, но при последующих компиляциях ссасhe существенно снижает время сборки. Особенно полезен для разработчиков, при сборках больших файлов, так как можно выставить в настройках объем кеша.

Трудности с которыми столкнулся. При установке нового ядра с отключенными модулями и символьной информации очень тормозило систему, поэтому я в загрузчике GRUB загружался со старого ядра.