

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт: КБСП

Направление: 09.03.02 «Информационные системы и технологии»

Кафедра: КБ-14 «Средства визуализации и обработки данных»

Дисциплина: «Методы Big Data»

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

**«Знакомство с Hadoop, MapReduce и файловой системой HDFS»**

Выполнил студент БСБО-08-18

Шайко Т.И.

Руководитель работы: Лебедев А.С.

Москва 2021 г.

# **Оглавление**

[Теоретическое введение 3](#_Toc86180505)

[Цель работы 4](#_Toc86180506)

[Задача работы 4](#_Toc86180507)

[Ход выполнения 5](#_Toc86180508)

[Запускаем пример «WordCount» 9](#_Toc86180509)

[Подготовка 9](#_Toc86180510)

[Некоторые детали 13](#_Toc86180511)

[Запускаем и тестируем код из «WordCount» 17](#_Toc86180512)

[Заключение 25](#_Toc86180513)

[Список используемой литературы 26](#_Toc86180514)

# Теоретическое введение

**Hadoop** — это фреймворк, предназначенный для распределённого хранения в серверных кластерах данных больших объёмов (BigData) и их обработки методом MapReduce. На его появление очень сильно повлияла компания Google; Hadoop вышел из так называемого проекта Nutch и потом успешно развивался самостоятельно как отдельный большой проект.

«Собственно, Hadoop — это распределенная архитектура «главный-подчиненный», которая состоит из распределенной файловой системы Hadoop (**HDFS**) для хранения и **MapReduce** для обеспечения вычислительных возможностей. Характерными чертами Hadoop являются разделение данных и параллельные вычисления над большими наборами данных. Его хранилище и вычислительные возможности масштабируются с добавлением узлов в кластер[[1]](#footnote-1), и могут обрабатывать петабайты данных в кластерах с тысячами улов.

HDFS — это компонент хранения Hadoop, распределенная файловая система, вдохновленная статьей о файловой системе Google (GFS). HDFS оптимизирована для обеспечения высокой пропускной способности и лучше всего работает при чтении и записи больших файлов (гигабайт и больше). Для поддержки этой пропускной способности HDFS использует необычно большие (для файловой системы) размеры блоков и оптимизацию локализации данных для уменьшения сетевого ввода/вывода.

Масштабируемость и доступность также являются ключевыми характеристиками HDFS, что отчасти достигается за счет репликации данных и отказоустойчивости. HDFS реплицирует файлы заданное количество раз, устойчив к программным и аппаратным сбоям, а также автоматически реплицирует блоки данных на вышедших из строя узлах» [1, с.5].

Технология Hadoop’а сейчас активно используется везде, где есть большие объёмы данных — в гигантах, типа, Facebook, Yahoo, Amazon, Twitter и многих других.

# Цель работы

Установить Hadoop и всё, что с ним связано, и рассмотреть пример применения модели MapReduce на практике — в программе-примере «WordCount».

# Задача работы

1. Скачать образ CentOS 7 и установить его на виртуальную машину;
2. Скачать и установить JDK;
3. Скачать и установить архив с библиотеками Hadoop;
4. Настроить SSH-подключение к машине с CentOS;
5. И настроить Hadoop, установив скрипты, выданные преподавателем;
6. А также протестировать работу этого инструмента на примере «WordCount» подсчёта кол-ва слов в документах.

# Ход выполнения

Сначала мы установим ОС, основанную на Линуксе — CentOS 7 — себе на виртуальную машину.

Затем скачаем JDK (Java SE Development Kit 8) и установим его на нашу новую систему для возможности работать с Java-кодом.

После этого скачаем с выданной преподавателем ссылки Hadoop второй версии (рис. 1).



Рисунок 1 — Этап установки необходимого ПО для последующего выполнения лабораторных работ

И устанавливаем его.

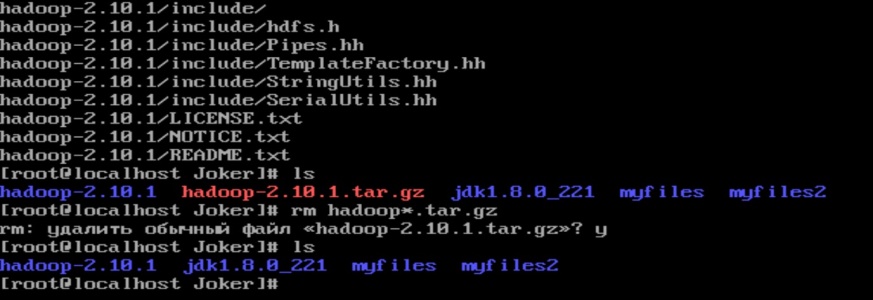


Рисунок 2 — Две только что установленные папки в домашнем каталоге пользователя «/home/Joker»

Следующим шагом мы настраиваем переменные окружения нашей системы (рисунок 3). Добавляем переменные «JAVA\_HOME» и «HADOOP\_HOME», которые определим чуть позднее.

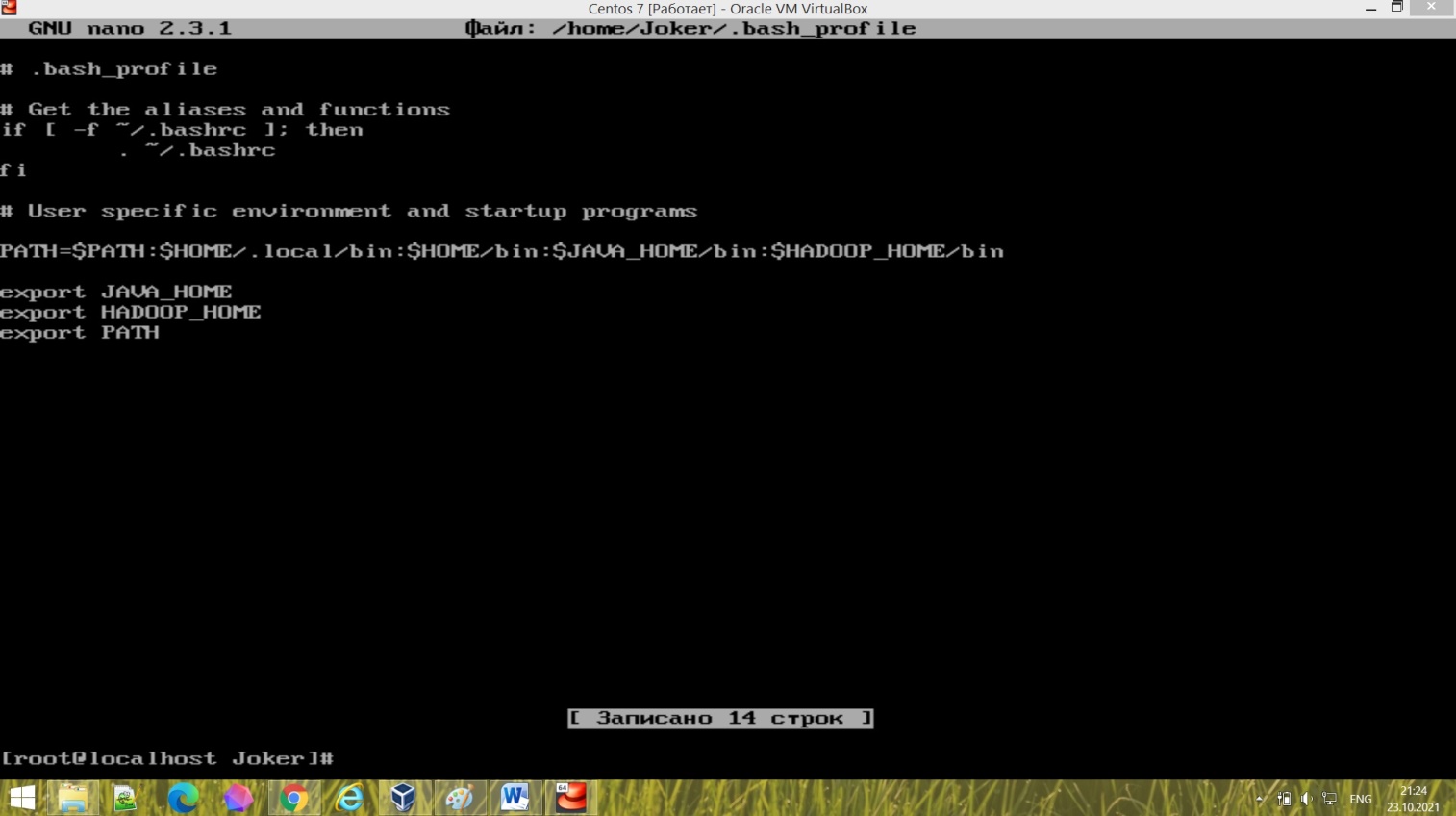


Рисунок 3 — Настройка переменных окружения нашей CentOS

Потом настраиваем SSH-подключение к нашей машине c CentOS.

Дальше мы настраиваем некоторые конфигурационные файлы в каталоге «/home/Joker/hadoop-2.10.1» для удобства. Изменяем их, прописав несколько новых .xml-тегов (*etc/hadoop/yarn-site.xml*, *etc/hadoop/mapred-site.xml*, *etc/hadoop/hdfs-site.xml* и другие).

По заданию нам нужно посмотреть документацию[[2]](#footnote-2) Hadoop’а и выяснить, что за это за такие конфигурационные файлы. Скачиваем для этого браузер к себе на CentOS под названием «links» и через него смотрим документацию. Чтобы было удобнее, можно скачать к себе на основную ОС «putty» — SSH-клиент, через который как раз и можно включить установленный на CentOS браузер «links» и полистать этот сайт с документацией (который хранится локально на виртуалке в скачанном нами ранее каталоге «hadoop-2.10.1»), используя более удобный курсор мышки (рисунок 4).

А можно сделать проще — воспользоваться своим нормальным браузером, посетив сайт «Хадупа» по ссылке «https://hadoop.apache.org/docs/r2.10.1/».

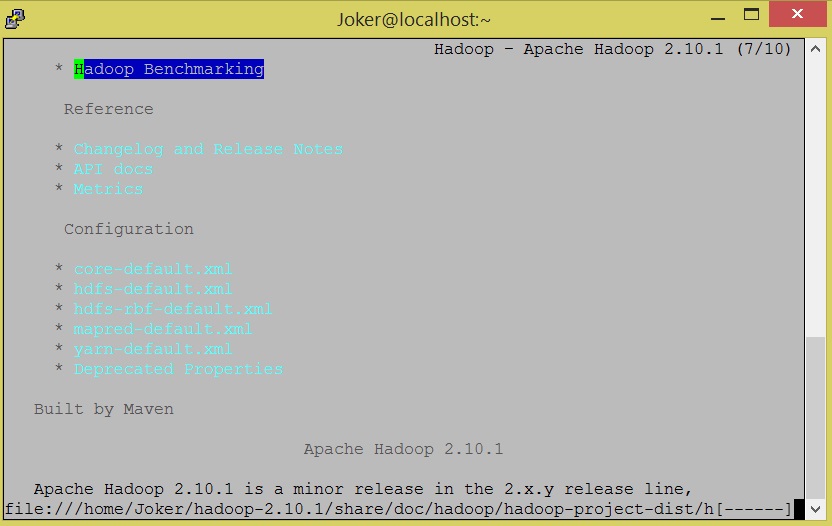


Рисунок 4 — Браузер «links», установленный на нашу CentOS 7 и открытый в «Putty».

Прочитав быстренько содержимое, узнаём назначение прописанных в конфигурационные файлы (см. чуть выше) параметров. К примеру, параметр «fs.defaultFS» отвечает за определение файловой системы для «Хадупа» по умолчанию. Кроме этого, устанавливая этот параметр, мы даём понять, какой адрес хоста, порт и т.д. мы используем при подключении к «Хадупу».

Перемещаем каталоги с «Хадупом» и «JDK» в домашний каталог, как сказано в задании в самом начале (лучше они пусть лежат там — по заданию, а то мало ли). И отредактируем файл «~/.bach\_profile», как мы это сделали ранее для файла «/home/Joker/.bach\_profile» (см. рисунок 3). И, если необходимо, перенастраиваем и SSH для домашней директории по точно такой же схеме, что и раньше.

Заключительным шагом скачиваем у преподавателя его скрипты .sh и засовываем их в папку «~/bin» (см. рисунок 5).

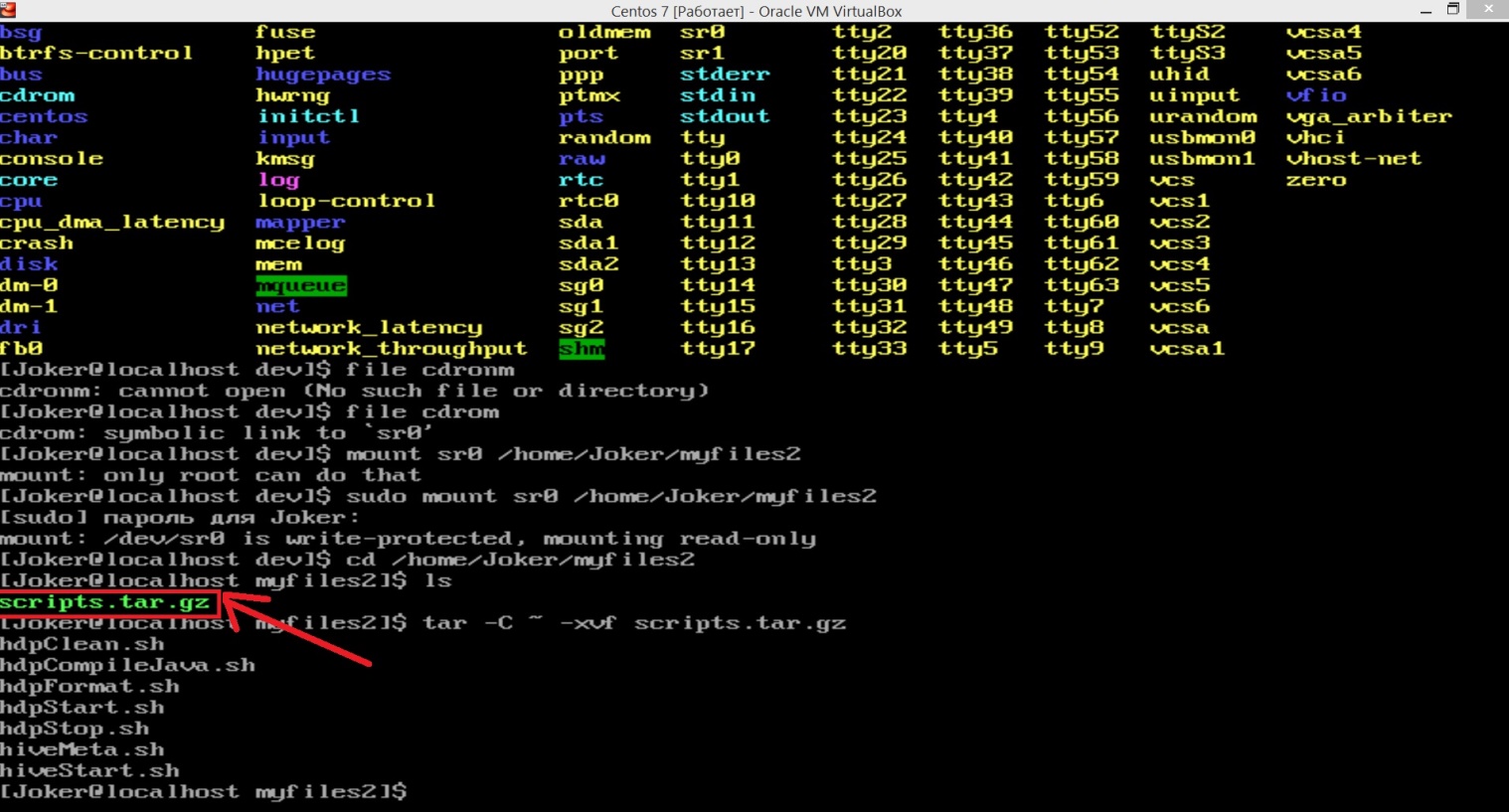


Рисунок 5 — Перенос архива со скриптами преподавателя с внешнего диска на нашу CentOS, распаковка этого архива и перенос в каталог «~/bin»[[3]](#footnote-3) (у меня криво работал «wget»).



Рисунок 6 — Итоговый результат настройки «Хадупа»

## Запускаем пример «WordCount»

### Подготовка

Сначала установим к себе на виртуальную машину с CentOS архив с примером, а потом распакуем его в домашний каталог (рисунок 7).

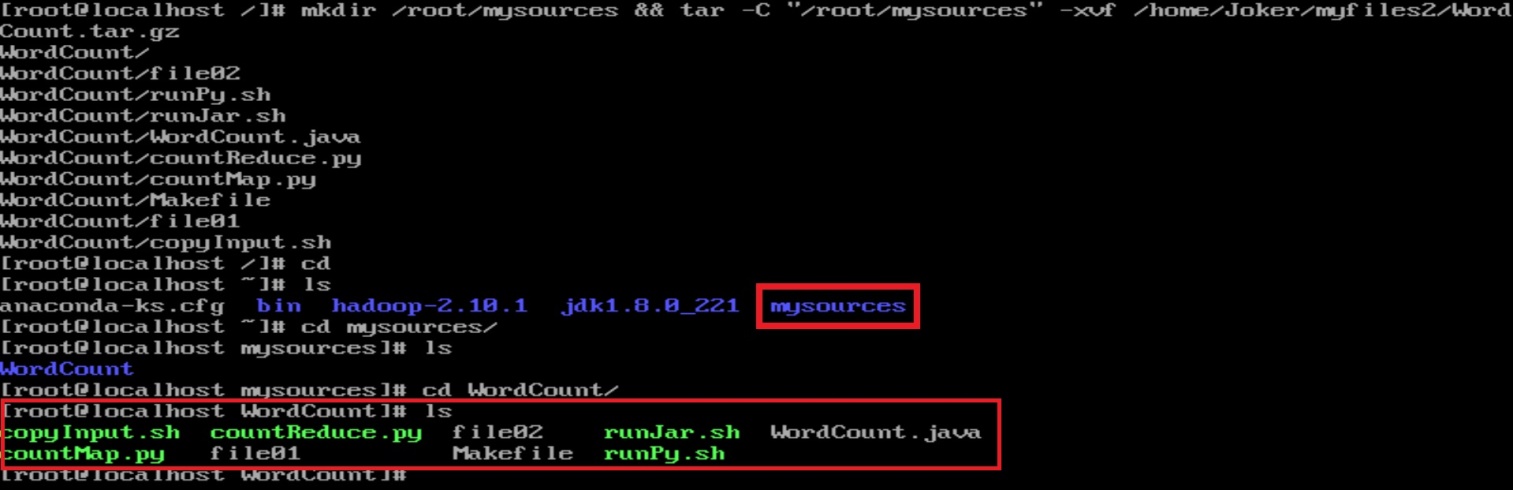


Рисунок 7 — Установка к себе на машину файлов с примером «WordCount»

Теперь форматируем нашу файловую систему HDFS:

#hdfs namenode –format[[4]](#footnote-4)

(утилиту «hdfs» можно найти в каталоге «*/root/hadoop-2.10.1/bin*»[[5]](#footnote-5)).

И запускаем сервисы Hadoop (см. рисунок 8): переходим в папку «*/root/hadoop-2.10.1/sbin*» и запускаем на выполнение скрипт «start-dfs.sh»:

#cd /root/hadoop-2.10.1/sbin && ./start-dfs.sh

Потом снова переходим к «hdfs» и говорим интерпретатору bash исполнить следующие команды (это всё в рамках включения и подготовки «Хадупа»):

#cd /root/hadoop-2.10.1/bin && hdfs dfs –mkdir /user

#hdfs dfs –mkdir /user/`id –nu`

Код 1 — Переход в директорию с «hdfs» и создание новой директории в нашей распределённой файловой системе — «*/user*»; затем создание вложенной директории «*/user/root*» («*id –nu*» вернёт «*root*» в нашем случае)



Рисунок 8 — Скрипт «hdpStart.sh»

В случае если при запуске кода 1 у нас появилось сообщение, наподобие «*hdfs: command not found*», то нам нужно изменить переменную среды «PATH», зайдя в текстовый редактор, к примеру «nano», и отредактировав скрытый файл «*/root/.bash\_profile*». Нужно просто добавить туда конкретные пути к важнейшим утилитам, таким как: «*hdfs*», «*yarn*», «*start-dfs.sh*», «*start-yarn.sh*» и другим. К примеру, путь до первых двух утилит — «*/root/hadoop-2.10.1/bin*» — опять же в моём случае.

После выполнения этих команд (именно в таком порядке — иначе у тебя не получится) мы сможем увидеть созданные в нашем datanode[[6]](#footnote-6) каталоги «*/user*» и «*/user/root*» (рисунок 9).

Потом пишем и исполняем оставшиеся две команды запуска скриптов «Hadoop» (рисунок 10).

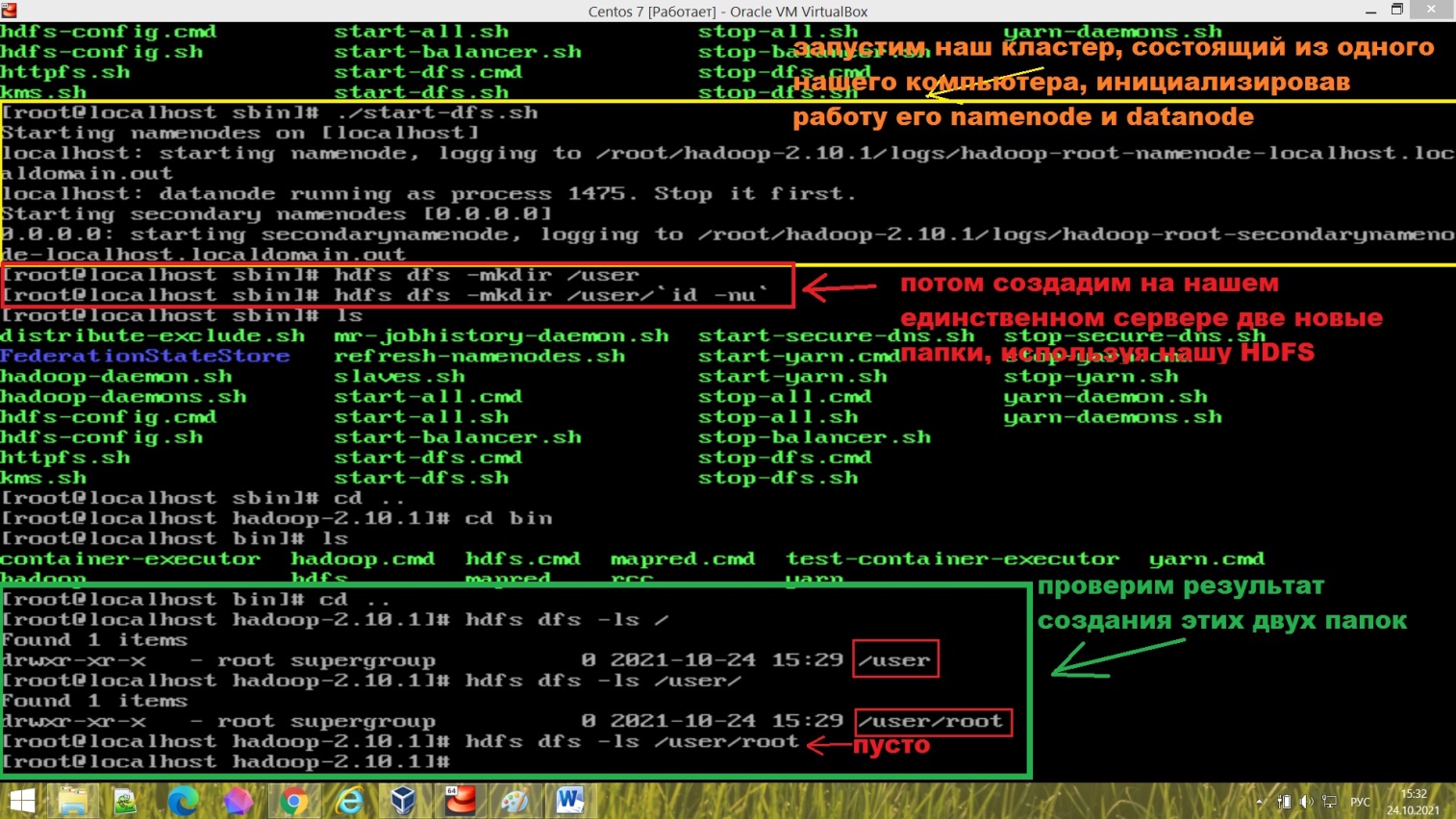


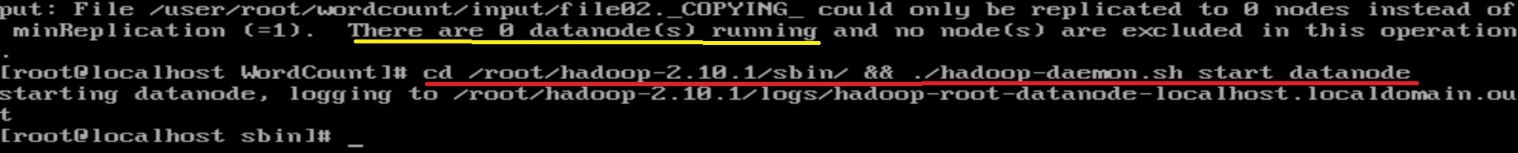
Рисунок 9 — Результат выполнения наших первых команд из скрипта на рис. 8



Рисунок 10 — Исполнение ещё двух встроенных скриптов «Хадупа»

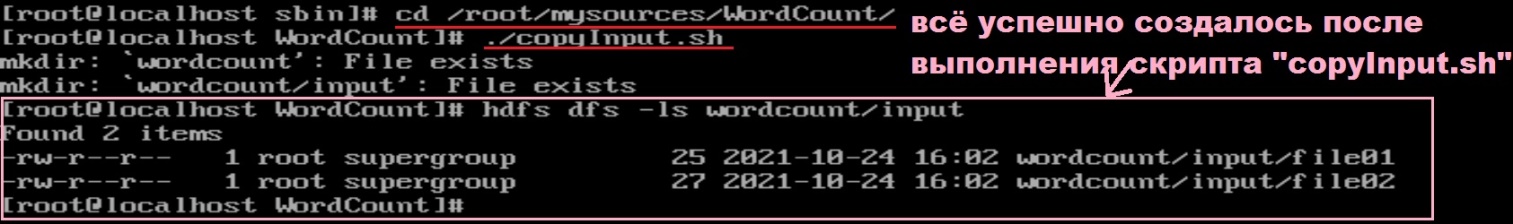
Затем нам нужно запустить скрипт, лежащий в каталоге с примером «WordCount». Он называется «*copyInput.sh*».

Если ты был внимателен, то мог заметить, что datanode нашего сервера не удалось запустить: это можно увидеть в сообщении на рисунке 9 в жёлтой рамке — «*localhost: datanode running as process 1475. Stop it first*». Из-за этого появится ошибка при исполнении скрипта «*copyInput.sh*» (текст ошибки подчёркнут жёлтой линией):



Значит, нам нужно просто запустить его командой с красной линией.

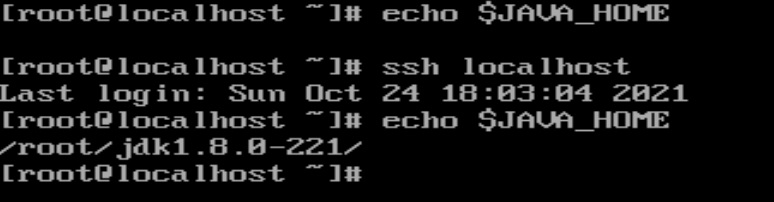
После этого всё будет Окей:



### Некоторые детали

**Файл .bash\_profile**

Файл «*/root/.bash\_profile*», который мы используем для определения переменных окружения, таких как, «JAVA\_HOME» или «HADOOP\_HOME» — действует только тогда, когда корневой пользователь — именно он — подключается по SSH. В других случаях эти наши переменные не будут определены:



К этому нужно быть внимательным! [4]

**Ctrl+E**

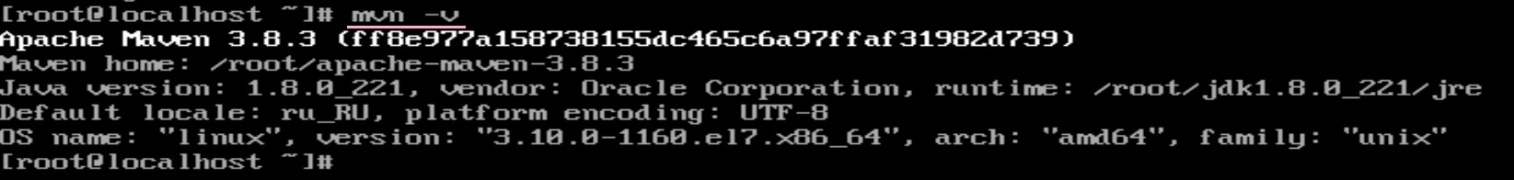
И ещё одно напоминание: переместиться в конец строки в редакторе текста Линукса можно, нажав комбинацию клавиш «Ctrl+E».

**Maven**

Перед компиляцией «*WordCount.java*» в «*WordCount.jar*» нам нужно подключить библиотеки «*org.apache.hadoop*», то есть библиотеки «Хадупа». Иначе, будет выдаваться ошибка синтаксиса и компилированный файл «*WordCount.jar*» будет корявым и некорректным.

Для упрощения локального подключения java-библиотек к себе в проект «Apache» придумали инструмент «*Maven*». Он с помощью файла «*pom.xml*», лежащего в твоём проекте, скачивает прямо из Интернета и подключает все нужные тебе библиотеки. Нужно только указать с помощью языка разметки POM — подмножества известного языка XML — название, ссылку, версию библиотеки и по желанию другие параметры её установки. [5]

После скачивания и установки по инструкции на официальном сайте «*Maven*» у нас запустится команда «*mvn*»:



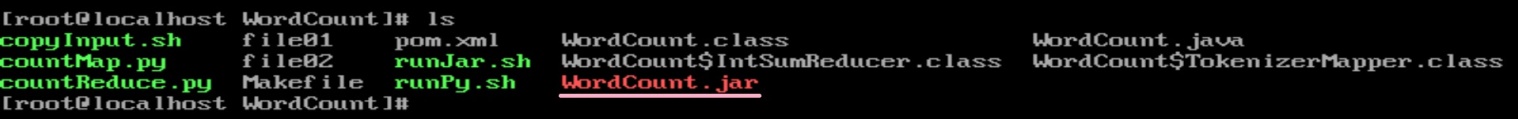
Это значит, что установка этого инструмента прошла успешно!

Но в принципе, посидев ещё некоторое время и подумав над этой ошибкой синтаксиса:

«*Класс WordCount не найден; Неизвестный символ №1 <…>; Неизвестный символ №2 <…>*»

— что означает непонимание компилятором выражений, наподобие «IntWritable» или «Text» — или, что то же самое, неудачное подключение к нам в проект библиотек «Хадупа» с помощью уже прописанного «*Makefile*». Я понял, что всё можно сделать проще — не устанавливать «*Maven*» и создавать нужный нам .jar-файл с его помощью, подключая все те же самые библиотеки «Hadoop»; а просто посмотреть на содержимое «*Makefile*» и указанного в нём скрипта «*hdpCompileJava.sh*» и найти именно в них недочёты — ведь у преподавателя всё работало именно при использовании утилиты «*make*», а не «*Maven*». И действительно, в «*hdpCompileJava.sh*» были прописаны библиотеки **версии 2.9.2**, а не установленной у меня **2.10.1**.

Исправив это, я смог успешно компилировать «*WordCount.java*» в «*WordCount.jar*» с помощью «*Makefile*»:



**firewalld в CentOS 7**

Это привычный нам в обычных операционных системах с графикой файрвол, или межсетевой экран. В этой работе я его использовал для пометки некоторых из портов (50070, 9000 и т.п.) как порты, входящие в доверенную зону (trusted): порты и интерфейсы, относящиеся к этой зоне у firewalld пропускают и выпускают абсолютно любой трафик (работают в ssh, dhcp и всех остальных доступных протоколах).

Также этот брадмауэр можно использовать для написания сетевых фильтрующих трафик правил.

Но применять этого защитника CentOS в нашей работе совсем необязательно!

**semanage в CentOS**

Эта утилита используется в специальной системе безопасности Линукса — SELinux (**S**ecurity-**E**nhanced Linux). В этой системе для определения прав доступа на какой-либо объект ОС используются специальные так называемые политики. SELinux действует поверх классической системы безопасности Linux, то есть с её помощью нельзя разрешить то, что запрещено основной системой безопасности Линукса.

SELinux работает в таких ОС, как: Ubuntu, CentOS, Debian, Fedora Core и в некоторых других.

Так вот утилита «semanage» используется для создания этих политик безопасности в отношении системы SELinux. Этот инструмент определённым образом фильтрует трафик, проходящий через интерфейсы и порты.

Изначально утилита знает, а значит и следит, не за всеми портами системы. Чтобы указать в правилах другие, нестандартные порты, нужно прописать их в этой команде. К примеру:

* Добавить новый порт в систему SELinux:

# **semanage** port -a -t http\_port\_t -p tcp 6322

* Удалить порт:

# **semanage** port -d -t http\_port\_t -p tcp 2223-2225

Некоторые важные для всей нашей ОС порты удалить из SELinux нельзя, к примеру порт №9000. Система должна особо наблюдать за подобными портами.

Этот инструмент я также использовал в работе — но делать это необязательно. Проворачивай такие штуки только тогда, когда тебе действительно это нужно.

**DNF и YUM**

Первая утилита используется для скачивания из официального репозитория программных пакетов и библиотек и для их установки. Вторая делает то же самое.

YUM работает в CentOS 7, а DNF — в CentOS 8. Последний является более новым инструментом. Он написан не только на Питоне, как первый — YUM, а также и на Си и на C++. Следовательно, DNF более быстрый и поддерживает более обширное взаимодействие с другими программными модулями; также его легче изменить, добавив свои или даже удалив некоторые его компоненты (YUM изменить сложнее также из-за неполной документации).

Но YUM все равно хороший инструмент **:)** .

### Запускаем и тестируем код из «WordCount»





На рисунках выше можно видеть тестирование работы «MapReduce» «Хадупа». И его результат.

Так как мы указали два потока для операции «Reduce» (рисунок 11) и перенаправили к нам на экран выход только одного из них, то у нас в консоли мы увидим неполную информацию о результате работы программы (не хватает отчёта о ещё двух словах в «*file01*» и «*file02*» — «Hello» и «World» — составлением которого занимался второй «Reduce»). Можно для удобства воспользоваться Web-интерфейсом, который позволит нам просмотреть наглядно, а не через консоль, содержимое файловой системы HDFS нашего маленького серверного кластера и увидеть полный отчёт о работе программы «WordCount» в более читаемом и визуализированном виде (но можно также пользоваться и консолью — как кому удобнее).

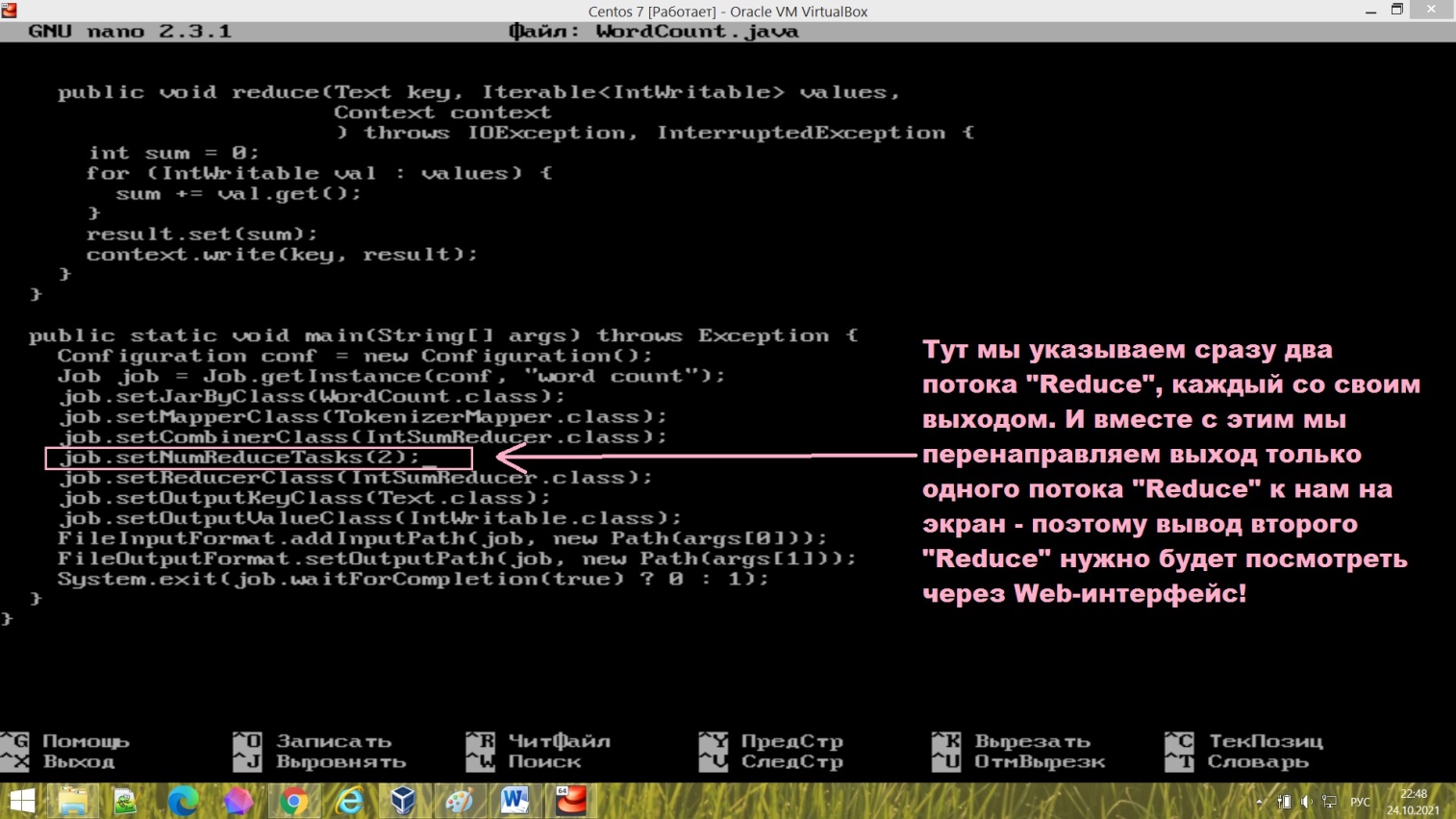


Рисунок 11 — Код основного исходника примера — «WordCount.java»

Мы пойдём к браузеру для разнообразия нашей уже немного надоевшей «консольной жизни». Для того чтобы получить полный доступ для просмотра файловой системы HDFS через наш браузер по HTTP, мы можем указать в файле «*/root/hadoop-2.10.1/etc/hadoop/core-site.xml*», который мы модифицировали ранее (см. раздел «Ход выполнения» страницу №6), ещё одно свойство под названием «*hadoop.http.staticuser.user*». Задаём этому свойству значение имени пользователя, установившего «Hadoop» — у нас это корневой пользователь «root» (рисунок 12).

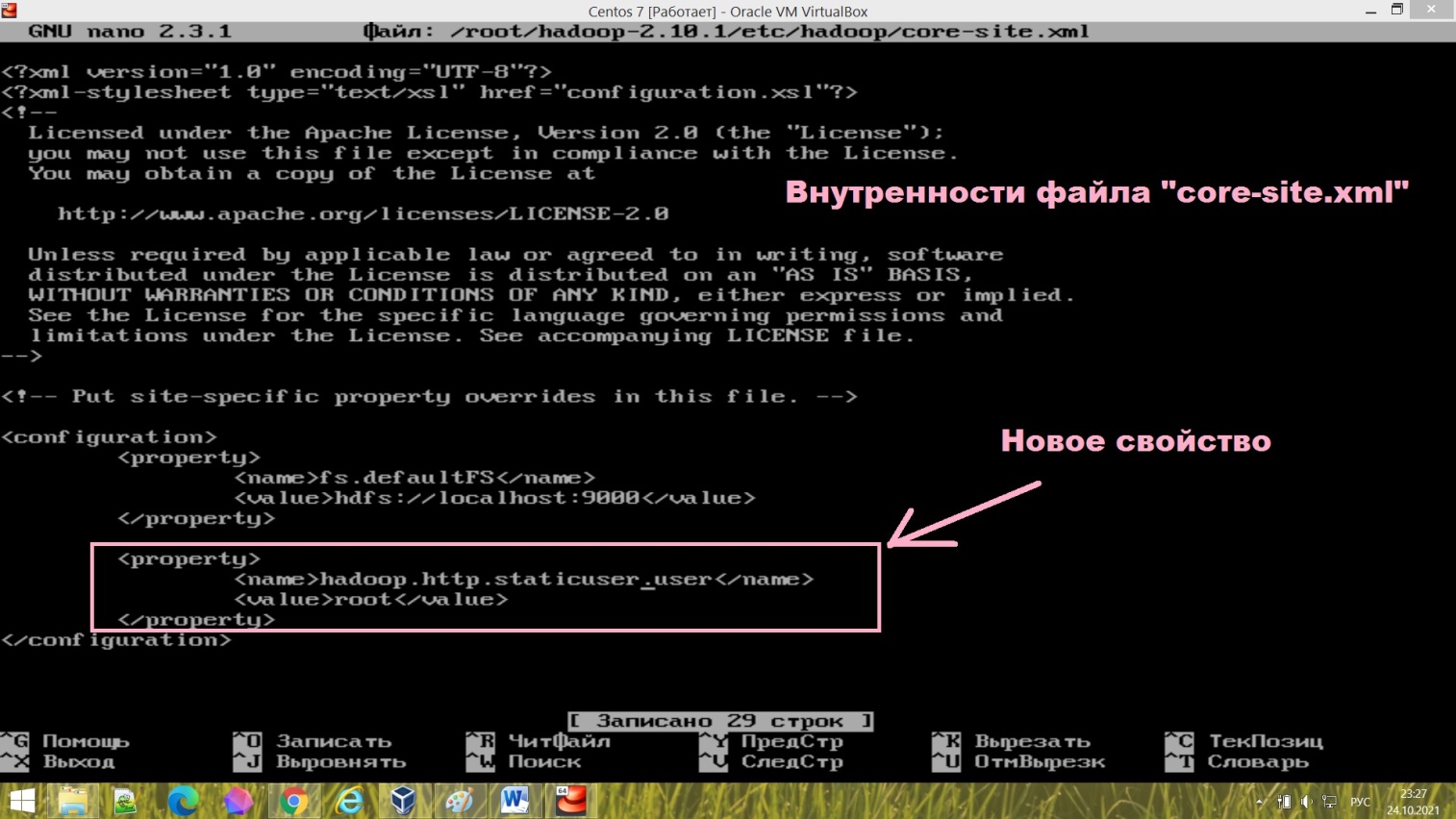


Рисунок 12 — Изменённый файл «*core-site.xml*»

Консольные браузеры отображают только совокупность и примерное местоположение всех ссылок на веб-странице и не являются полноценными браузерами, поэтому мне не удалось просмотреть в более наглядном виде итоговые результаты через консольный браузер.

Откроем под SSH новый порт для того, чтобы подключиться с нашей основной ОС Windows 8 к виртуальной машине с CentOS 7.

Сначала настраиваем файрвол виртуальной машины, добавляя в список открытых портов порт с номером к примеру 50070 и перезагружая его:

#firewall-cmd --permanent --zone=public --add-port=50070/tcp

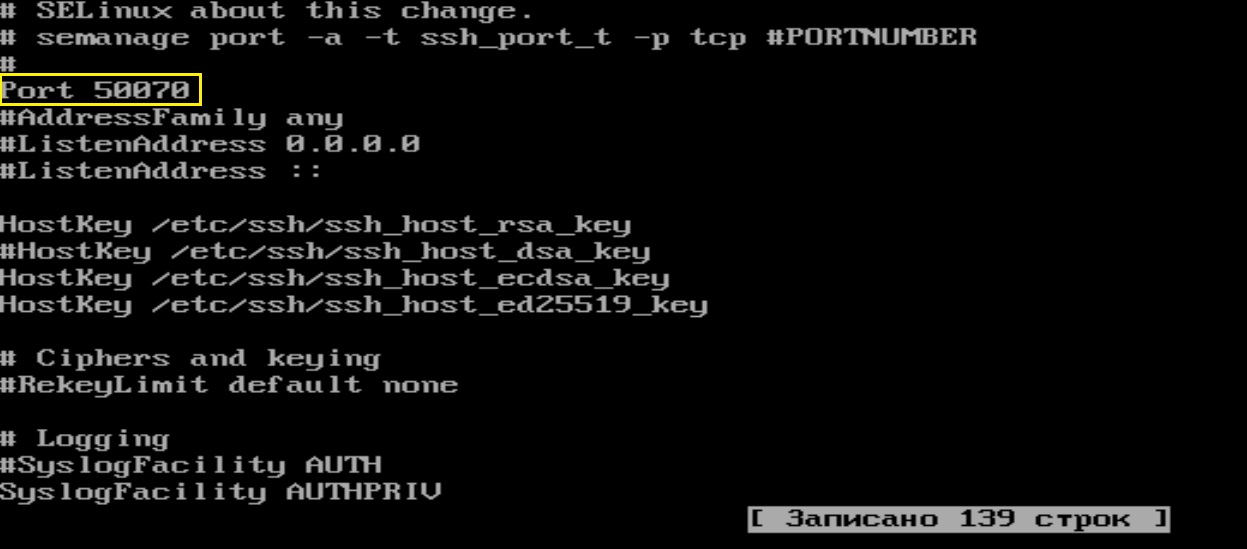
#firewall-cmd --reload

Дальше нам нужно редактировать уровень доступа к этому порту: установить для него так называемые правила контроля доступа SELinux с помощью утилиты «*semanage*»:

#semanage port -a -t ssh\_port\_t -p tcp 50070

Ключ «-a» говорит о добавлении в правила SELinux нового порта; «-t» говорит нам о типе порта системы SELinux (есть также тип «http\_port\_t» и другие); а параметр же «-p» указывает нам на используемый протокол передачи данных (у нас tcp).

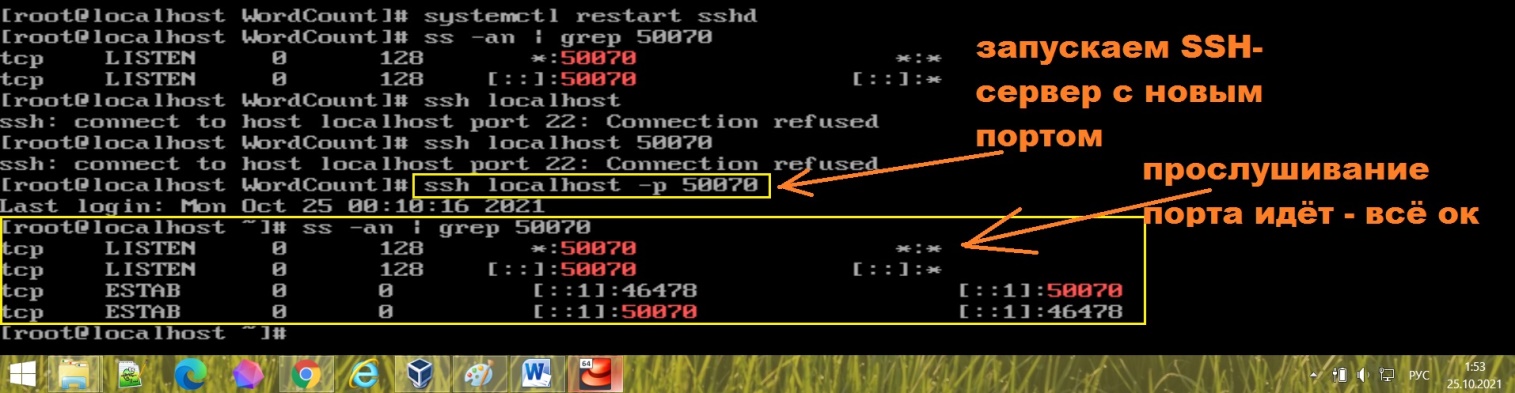
После этого нам нужно настроить SSH, изменив файл «*/etc/ssh/sshd\_config*»: необходимо раскомментировать строчку «*#Port 22*» и изменить номер порта на 50070:



И заключительно перезапустим службу SSH в CentOS — «*sshd*»:

#systemctl restart sshd

Теперь запустим наш SSH-сервер с новым портом №50070 и проверим работу службы:



Всё! Порт открыт[[7]](#footnote-7) — мы можем смело подключаться к «Хадупу» через него. Смотри рисунок 13.

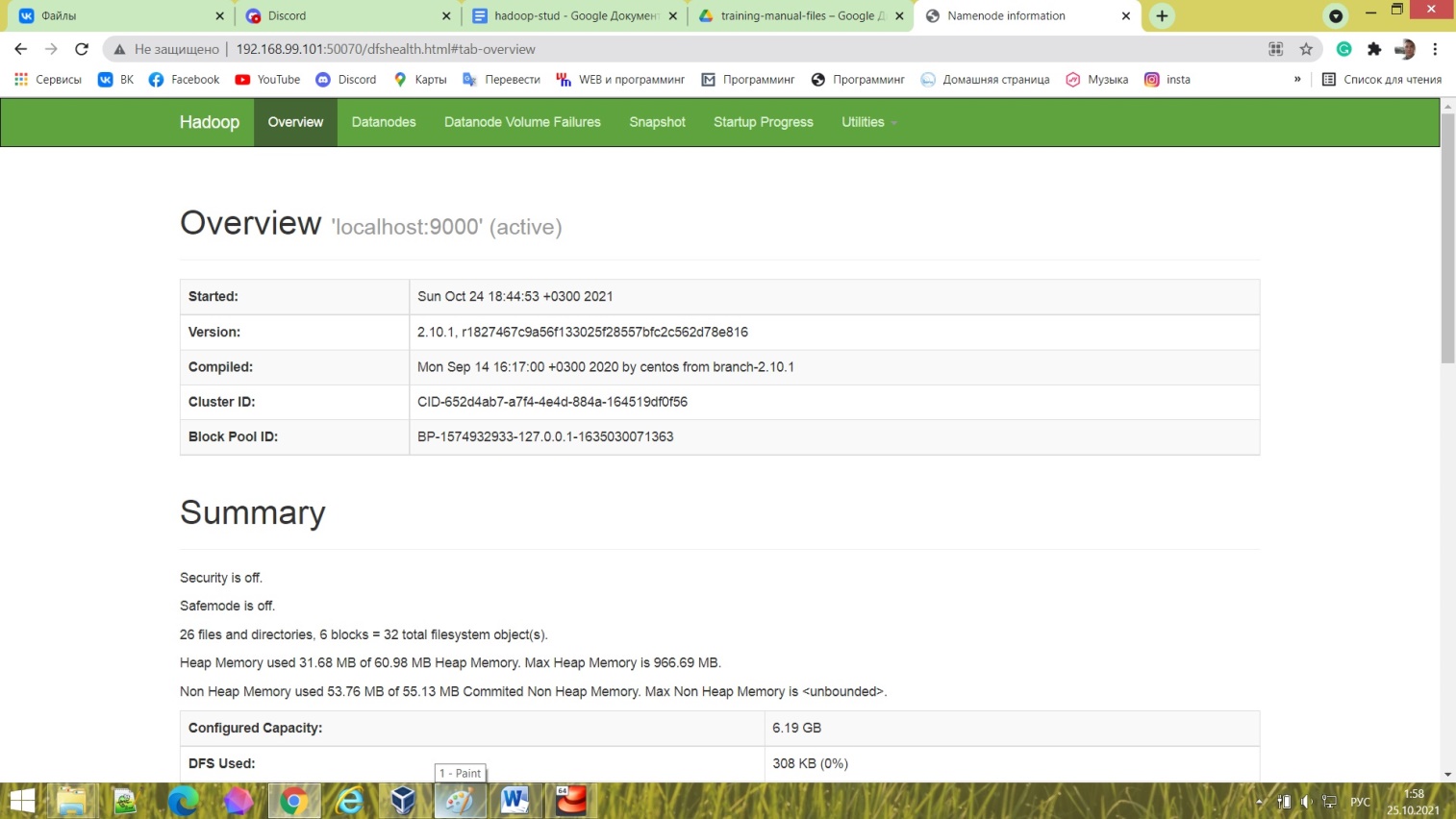
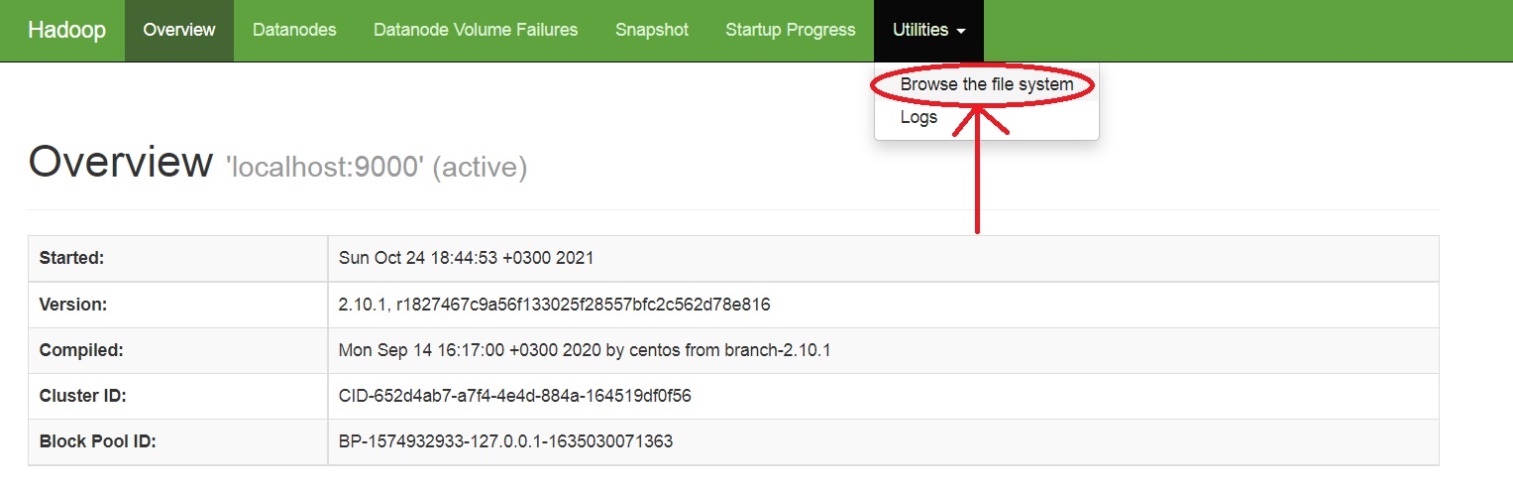


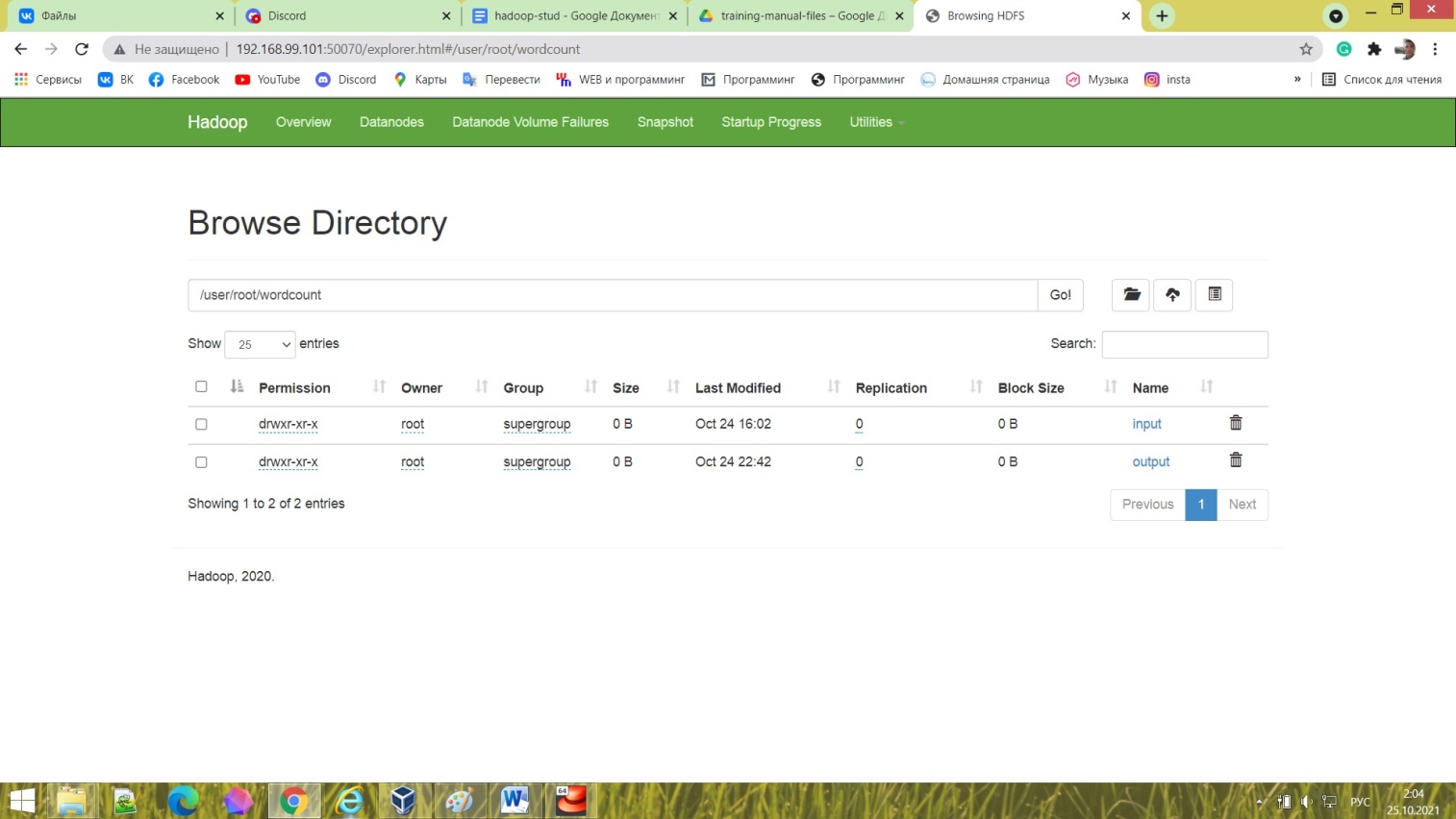
Рисунок 13 — Удачное подключение к HDFS Hadoop’а!

На главной странице мы можем видеть основную информацию о работе «Хадупа»: кол-во работающих узлов кластера, занятую и свободную память, версию «Хадупа», id кластера, время последних действий разных программ в HDFS и многое другое.

Теперь выбираем в шапке сайта раздел «*Utilities*» и жмём мышкой по пункту «*Browse the file system*»:

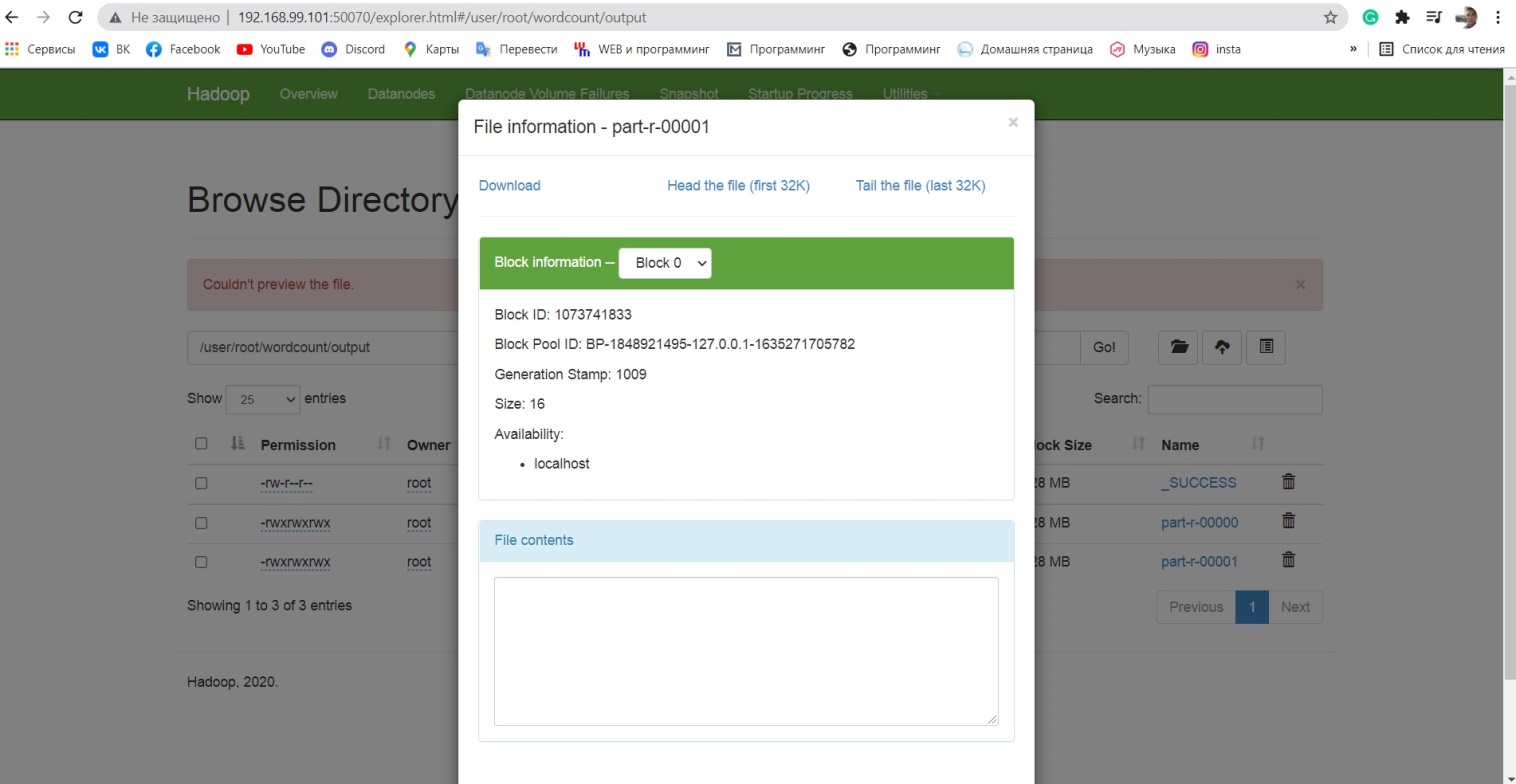


Перед нами откроется список директорий нашей распределённой файловой системы:

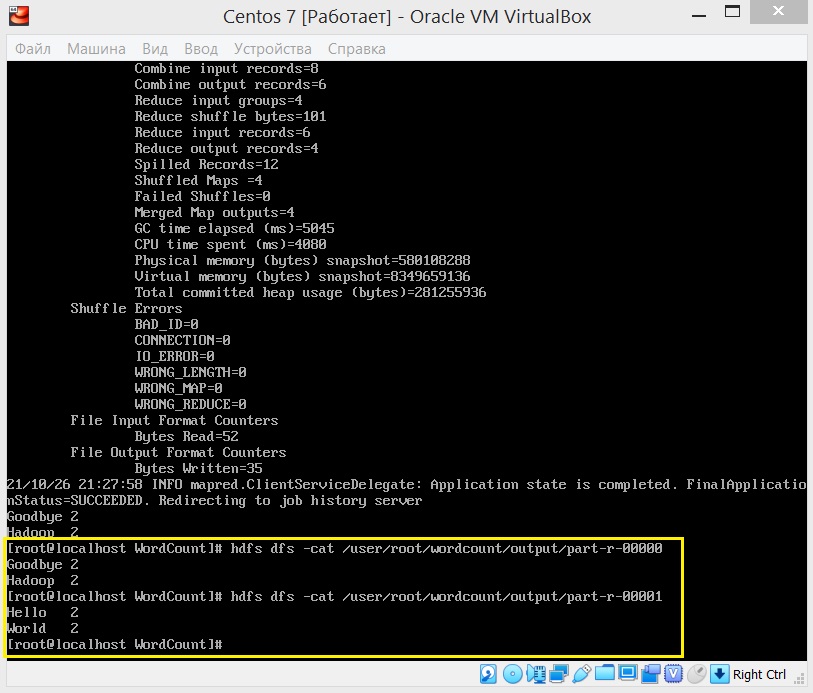


Заходим в показанную на рисунке выше папку и жмём по пункту «*output*», который система создала по завершении программы «WordCount». Мы с тобой увидим три файла, два из которых содержат в себе выходные данные двух созданных нами «Reduce»-потоков.

К сожалению, доступ к чтению этих файлов через WebUI мне не удалось получить. Для этого нужно в файле со свойствами «Хадупа» — «*$HADOOP\_HOME/etc/hadoop/core-site.xml*» указать пользователя-владельца «Хадупа». У меня это «*root*», но это не работает. Попробуй ты — может, у тебя получится **:)** .



Но можно посмотреть и через терминал:



Как видим, оставшиеся слова, «Hello» и «World», у нас посчитались верно!

На этом эта практическая работа по установке и тестированию «Hadoop» в части BigData заканчивается.

# Заключение

В ходе выполнения этой практической работы мы получили основные навыки работы с «Хадупом», поняли суть работы и предназначение этого инструмента, усвоили Java- и Python-способы написания и запуска кода с применением библиотек «Hadoop».

Конкретнее, мы разобрались с сущностью модели «MapReduce», научились устанавливать и редактировать свойства «Хадупа» в .xml-файлах, посмотрели на работу и отличия datanode и namenode, поняли, что такое yarn и многое другое.

В процессе выполнения работы я также познакомился дополнительно со многими полезными утилитами Linux (firewalld, semanage, maven и другими).

# Список используемой литературы

1. Методы Big Data [Электронный ресурс]: учебно-методическое пособие / А. С. Лебедев, Ш. Г. Магомедов. — М.: РТУ МИРЭА, 2021. — Электрон. опт. диск (ISO).
2. Hadoop // Википедия — Свободная Энциклопедия URL: https://ru.wikipedia.org/wiki/Hadoop (дата обращения: 23.10.2021).
3. Уайт Т. — Hadoop: Подробное руководство. — СПб.: Питер, 2013. — 672 с.: ил. — (Серия «Бестселлеры O’Reilly»).
4. LOSST // ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ В LINUX URL: https://losst.ru/peremennye-okruzheniya-v-linux (дата обращения: 24.10.2021).
5. Welcome to Apache Maven // Apache Maven Project URL: https://maven.apache.org/ (дата обращения: 24.10.2021).
6. 1. HDFS Ports // APACHE AMBARI PROJECT WEBSITE — AMBARI USER GUIDE URL: https://ambari.apache.org/1.2.3/installing-hadoop-using-ambari/content/reference\_chap2\_1.html (дата обращения: 26.10.2021).

1. горизонтальное масштабирование — одна из особенностей Hadoop’а [↑](#footnote-ref-1)
2. Документация находится локально в каталоге «hadoop-2.10.1» в виде сайта с начальной страницей «index.html». Также можно загрузить ту же самую документацию с помощью ссылки «https://hadoop.apache.org/docs/r2.10.1/» у себя в привычном браузере. [↑](#footnote-ref-2)
3. Чуть позже, исправляя свои недочёты, я перенесу файлы из этого архива в домашнюю директорию корневого пользователя, потому что работаю от его имени, а не от имени обычного пользователя «Joker» (см. рисунок 6)! [↑](#footnote-ref-3)
4. Скрипты, выданные преподавателем, у меня не запускаются из-за некорректной установки мной значений переменным окружения — поэтому делаем всё вручную! Эта команда есть в скрипте «hdpFormat.sh». [↑](#footnote-ref-4)
5. Я всё делал от имени пользователя «root»: установил все файлы в домашнюю директорию именно этого пользователя — в папку «/root/»! Смотри рисунок 6. [↑](#footnote-ref-5)
6. У нас в лабораторных работах будет простейший серверный кластер, состоящий из одного нашего компьютера — с системой CentOS. Этого будет достаточно для подобных работ. И вот этот наш единственный сервер содержит в себе данные в виде datanode’а и менеджер в виде namenode’а. [↑](#footnote-ref-6)
7. Как я выяснил позже, работать с портами нашего CentOS совсем необязательно — зайти с браузера твоей основной операционки получится и так. Нужно просто указать правильный сетевой ip-адрес твоей машины и протокол с портом для подключения к HDFS. Информацию про порты и протоколы «Хадупа» смотри на сайте с официальной документацией [6]. [↑](#footnote-ref-7)