Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Операционные системы»

Лабораторная работа №6  
«Работа с процессами в операционной системе Linux»

Студенты: Клоченко И.Е., Юсупов Д.А.

Преподаватель: Коротяев А.Н.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[1. **Вывод процессов** 3](#_Toc136637097)

[2. **Запуск и завершение процессов** 7](#_Toc136637098)

[3. **Работа с несколькими процессами** 10](#_Toc136637099)

[4. **Создание исполняемого файла** 12](#_Toc136637100)

[5. **Файлы Linux в сравнении с выводами команд** 13](#_Toc136637101)

[**Вывод:** 17](#_Toc136637102)

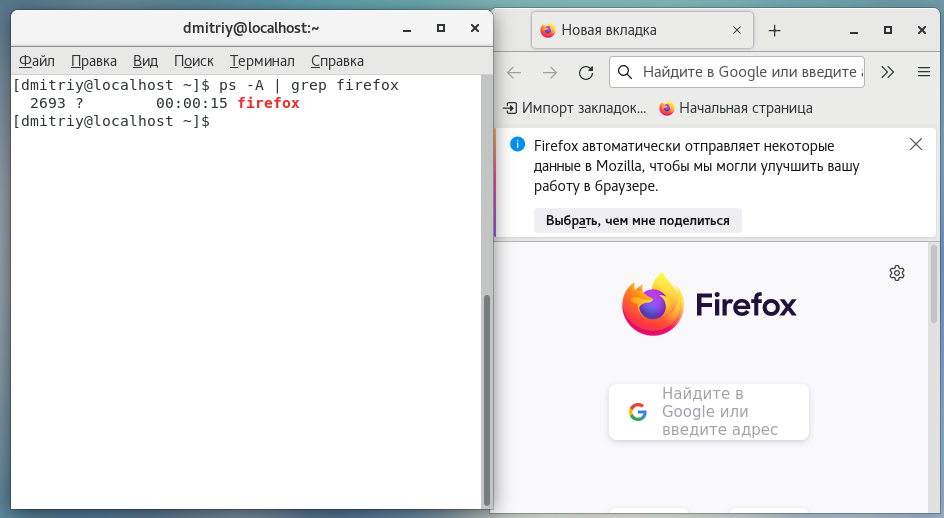
**Цель работы:** знакомство с организацией работы процессов в ОС Linux, изучение основных команд по работе с ними, получение практических навыков решения задач (к примеру, запуск и остановка процессов, запуск в режимах foreground и background – фоновый режим и непосредственный запуск).

**Ход работы:**

Комментарий: Лабораторная работа выполнена в CentOS 7, установленной на виртуальной машине VMWare Workstation.

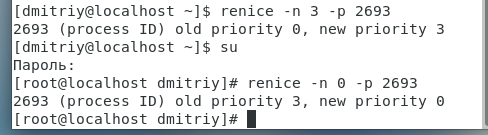
# **Вывод процессов**

При открытии браузера Mozilla Firefox запускается процесс *firefox* с уникальным PID (идентификатор процесса). Для вывода этого идентификатора на экран можно воспользоваться командой: *ps -A | grep firefox* (рисунок 1).



**Рисунок 1**. PID процесса firefox

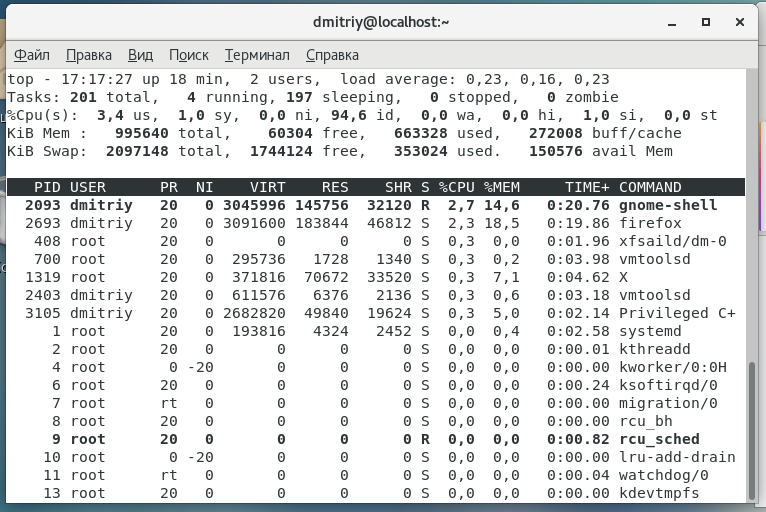
Для изменения приоритета процессов в ОС Linux есть утилита *renice -n <приоритет> -p <PID>*, при этом при понижении приоритета привилегии не требуются, а при повышении требуются права суперпользователя. По умолчанию приоритет равен 0, приоритеты распределены от -20 (наивысший приоритет) до 19 (наименьший приоритет). Для понижения приоритета процесса firefox можно воспользоваться командой: *renice -n 3 -p 2693* (рисунок 2).



**Рисунок 2**. Изменение приоритета процесса

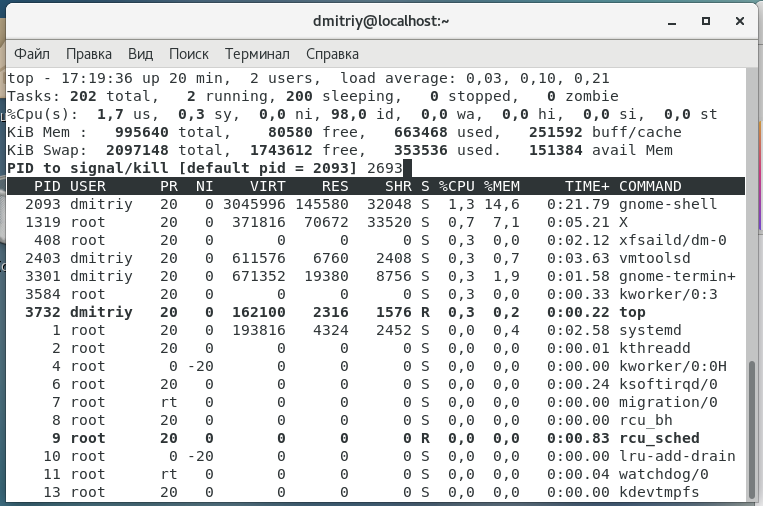
Для возврата приоритета 0 используется та же команда с другим параметром командной строки.

Утилита top предназначена для интерактивной работы с процессами (вывод, сортировка, завершение). Открывается командой *top*. На рисунке 3 процесс браузера является вторым сверху.

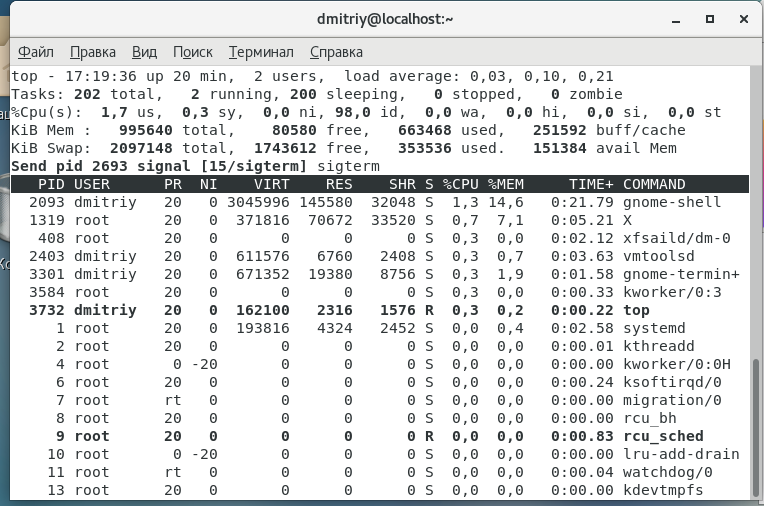


**Рисунок 3**. Утилита top

Для закрытия процессов в данной утилите необходимо нажать клавишу «k», ввести PID процесса и отправляемый сигнал (KILL, TERM и др.). Процесс закрытия браузера через утилиту top представлен на рисунках 4 и 5.



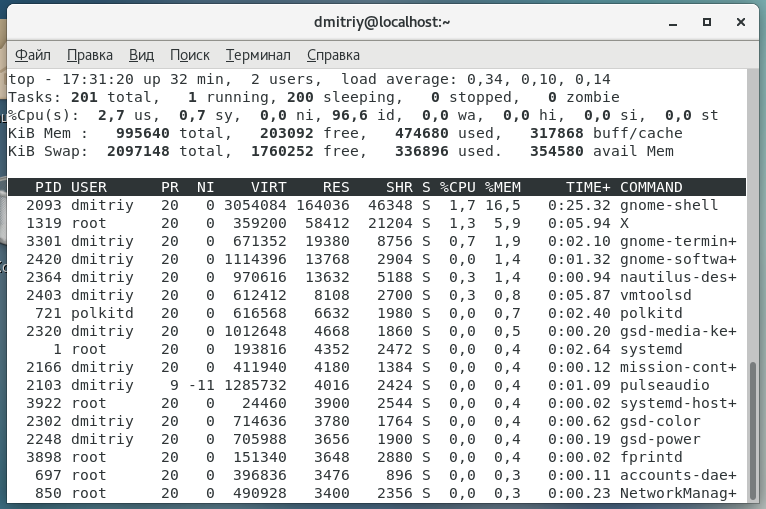
**Рисунок 4**. Завершение процесса с PID 2693



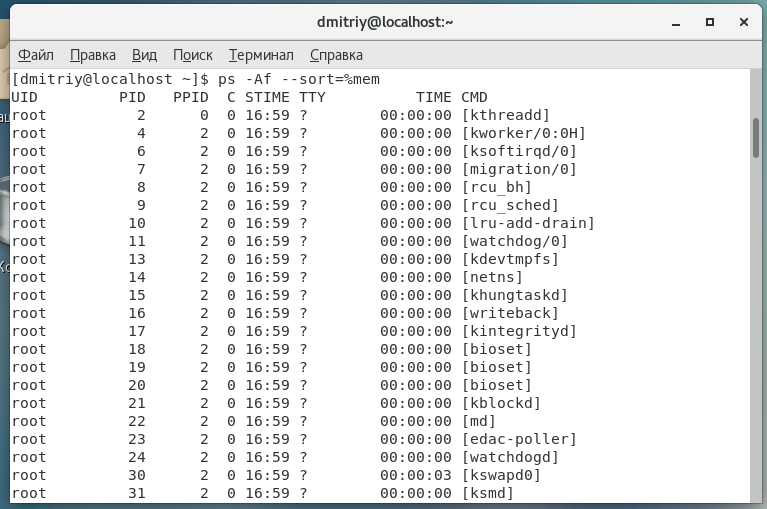
**Рисунок 5**. Выбор сигнала завершения процесса

Для вывода процессов, отсортированных по занятому объему памяти, можно воспользоваться двумя способами:

1. с помощью утилиты top нажатием сочетания Shift+M (интерактивный режим, выводит не все процессы) (рисунок 6);
2. командой *ps -Af --sort=%mem* (рисунок 7).

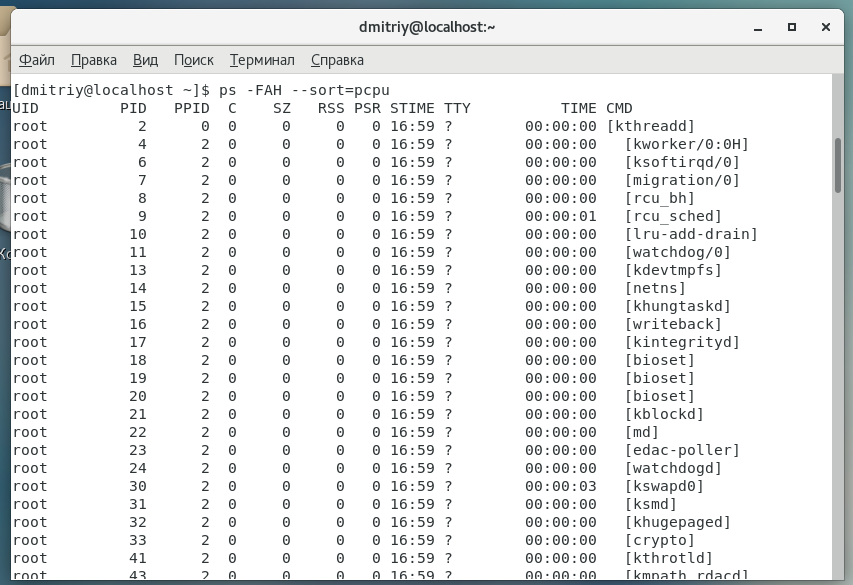


**Рисунок 6**. Сортировка процессов по занятому объему памяти в утилите top



**Рисунок 7**. Сортировка процессов по занятому объему памяти в утилите ps

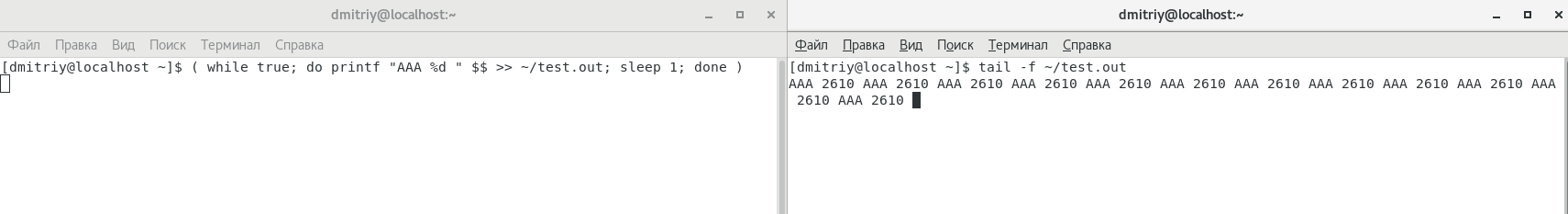
Для вывода процессов, отсортированных по используемым ресурсам процессора в виде дерева, можно воспользоваться командой: *ps -FAH --sort=pcpu* (рисунок 8) (-F – вывод большего количества информации о ресурсах, -A – вывод всех процессов в системе, -H – вывод процессов в виде дерева).



**Рисунок 8**. Процессы в виде дерева, отсортированные по использованию ресурсов процессора

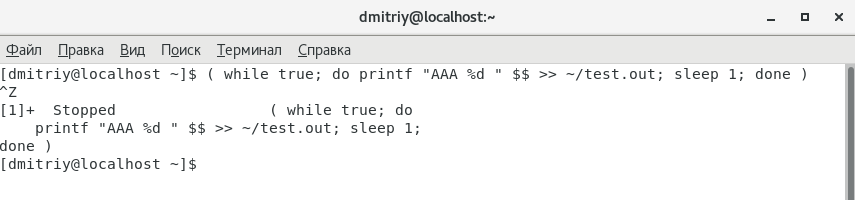
# **Запуск и завершение процессов**

Для запуска бесконечного процесса в системе можно воспользоваться бесконечным циклом. В левом окне терминала будет запущена команда: *(while true; do printf “AAA %d ” $$ >> ~/test.out; sleep 1; done)*, которая каждую одну секунду будет записывать в файл test.out в домашней директории пользователя строку “AAA <PID>”. В правом окне терминала для отслеживания записей в файл test.out можно воспользоваться командой *tail -f ~/test.out* (рисунок 9).



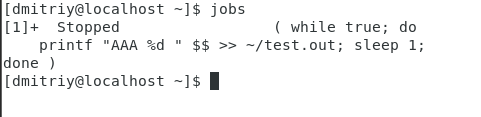
**Рисунок 9**. Бесконечный процесс и результат его работы

Сочетания клавиш Ctrl+Z позволяет послать сигнал остановки текущему процессу. При остановке бесконечного процесса его состояние с «Запущено» изменяется на «Остановлено», процесс перестает записывать в файл test.out и возвращает номер задания в квадратных скобках (в данном случае, задание 1) (рисунок 10).



**Рисунок 10**. Остановка бесконечного процесса

При выводе списка заданий (команда *jobs*) выводится та же информация, что и при остановке процесса (рисунок 11).



**Рисунок 11**. Список заданий jobs

Команды *bg* и *fg*:

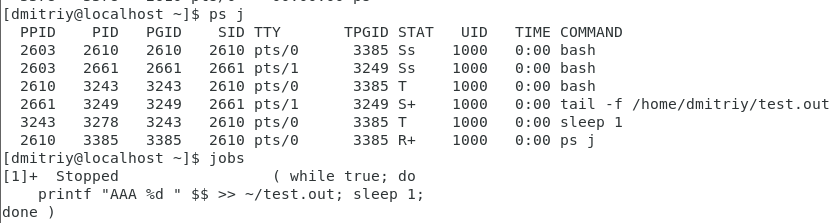
• fg <номер> переводит процесс на передний план (то есть продолжается явное выполнение);

• bg <номер> переводит процесс на задний план (то есть процесс продолжает работу в фоновом режиме);

• Знак «плюс» означает «текущее задание», т.е. то, с которым мы работаем сейчас. Знак «минус» означает «предыдущее».

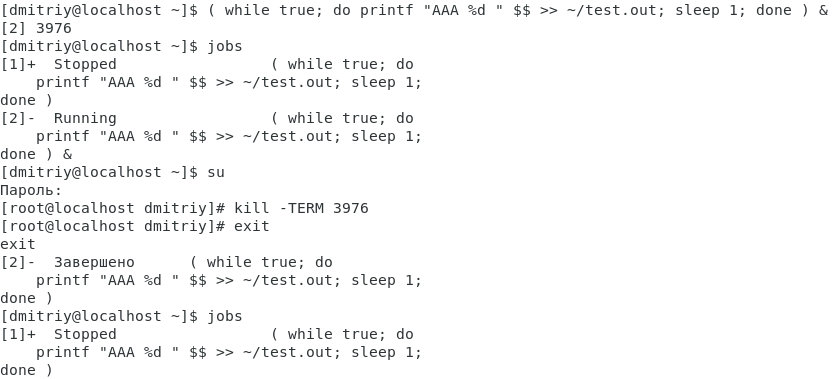
Команды *jobs* и *ps j* (рисунок 12):

* jobs выводит информацию о заданиях, выполняемых и остановленных;
* ps j выводит информацию о процессах в формате BSD, запущенных в различных терминалах.



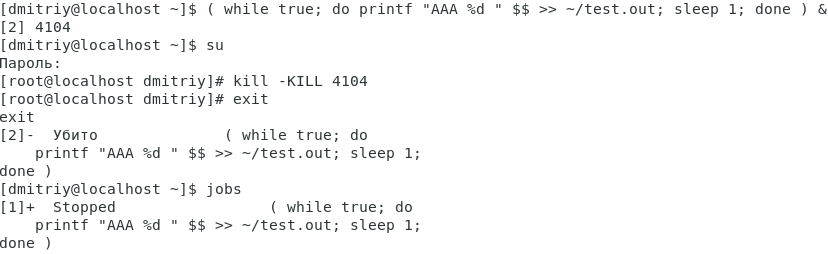
**Рисунок 12**. Команды *jobs* и *ps j*

При вводе в конце команды символа амперсанд (&) команда запускается в фоновом режиме (background) и выводится номер задания и PID процесса. После запуска бесконечного процесса в фоновом режиме запись в файл продолжилась. Состояние процесса при работе в фоновом режиме – «Запущено». При завершении процесса командой *kill -TERM <PID>* процесс завершается с состоянием «Завершено» и исчезает из списка задач jobs (рисунок 13).



**Рисунок 13**. Работа с процессом в фоновом режиме

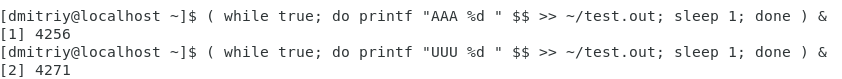
При повторении предыдущих действий с завершением процесса сигналом KILL изменяется только состояние процесса после завершения – «Убито» (рисунок 14). Стоит отметить, что для команды *kill* требуются права суперпользователя.



**Рисунок 14**. Завершение процесса сигналом KILL

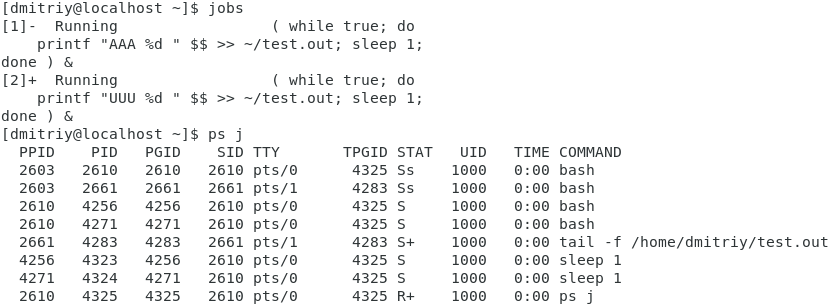
# **Работа с несколькими процессами**

Для запуска бесконечного процесса можно воспользоваться командой из предыдущего пункта: один процесс будет выводить AAA, второй процесс будет выводить UUU. Оба процесса будут запущены в фоновом режиме (с добавлением знака амперсанда в конце команды) (рисунок 15).



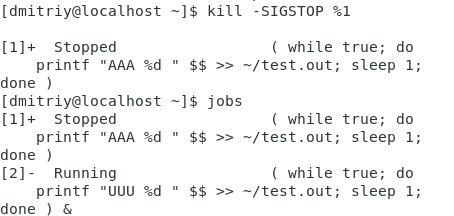
**Рисунок 15**. Два бесконечных процесса в фоновом режиме

При выводе списка заданий (*jobs*) и процессов (*ps j*) оба процесса имеют состояние «Запущено» (рисунок 16). Проведено наблюдение за работой процессов.



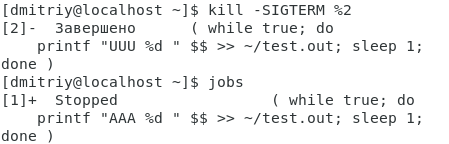
**Рисунок 16**. Список процессов и задач

Остановка первого процесса производится командой *kill -SIGSTOP %1*. При выводе списка заданий статус первого задания изменился на «Остановлено», в файл test.out перестали записываться строки типа AAA <PID> (рисунок 17).



**Рисунок 17**. Остановка первого процесса

Для завершения работы второго задания (сигнал TERM) необходимо воспользоваться командой *kill -SIGTERM %2*. При этом состояние процесса изменяется на «Завершено» и процесс исчезает из списка заданий *jobs* (рисунок 18).



**Рисунок 18**. Завершение работы второго процесса

Для завершения работы первого процесса возобновлю его выполнение командой *fg* и нажатием сочетания Ctrl+C (сигнал TERM). Завершение выполнения команды *tail* произведу аналогично.

# **Создание исполняемого файла**

Для создания файла сценария оболочки bash необходимо создать файл (команда *touch test-trap.sh*), внести в него код (редактор vim или nano, был использован vim: *vim test-trap.sh*) (рисунок 19). Код сценария:

*#!/bin/bash*

*declare -i i=0*

*trap 'echo "Аварийное завершение..."; exit ' SIGINT*

*while [ $i -lt 100 ]*

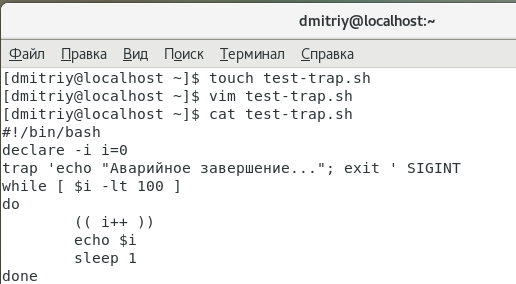
*do*

*(( i++ ))*

*echo $i*

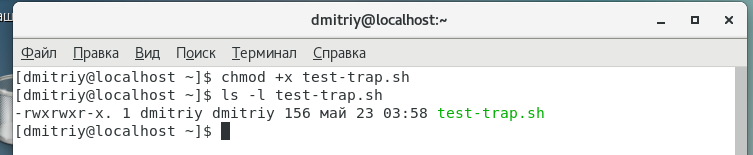
*sleep 1*

*done*



**Рисунок 19**. Создание файла сценария оболочки bash

Для возможности его запуска необходимо предоставить права запуска (execution) файлу, команда: *chmod +x test-trap.sh* (рисунок 20).



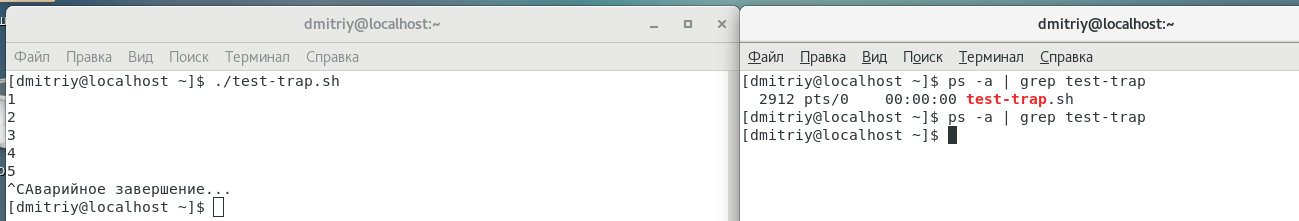
**Рисунок 20**. Добавление прав на запуск файла

При запуске файла в консоль выводятся числа, начиная с 1, каждую секунду увеличивающиеся на 1. При просмотре запущенных процессов с названием test-trap виден процесс с PID 2912. После нажатия сочетания клавиш Ctrl+C (сигнал SIGINT) программа перехватывает этот сигнал, выполняя команды:

*echo “Аварийное завершение…”*

*exit*

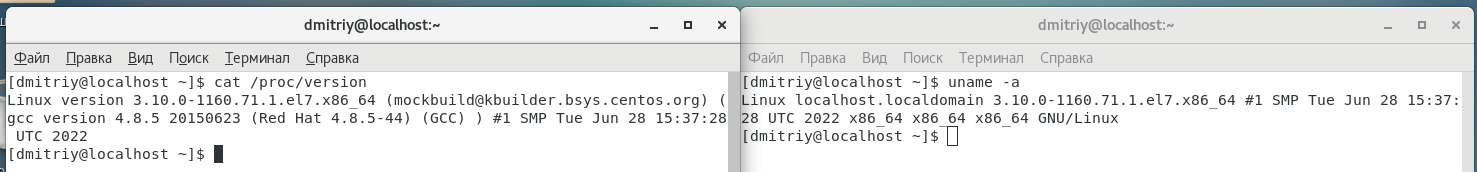
Процесс test-trap.sh исчезает из списка процессов. Все шаги представлены на рисунке 21.



**Рисунок 21**. Процесс test-trap.sh

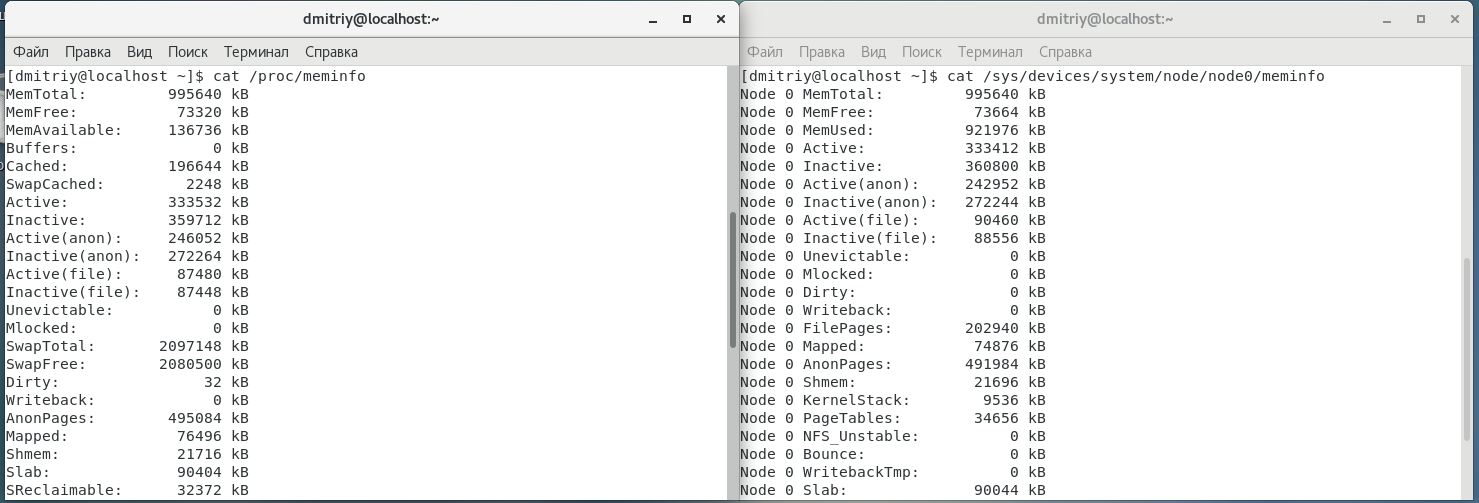
# **Файлы Linux в сравнении с выводами команд**

Файл */proc/version* содержит данные об версии используемого ядра Linux и версию gcc (GNU C compiler, компилятор C GNU). Команда *uname -a* выводит информацию об имени ОС, версии ядра и его имени, процессоре и аппаратных ресурсах (рисунок 22).



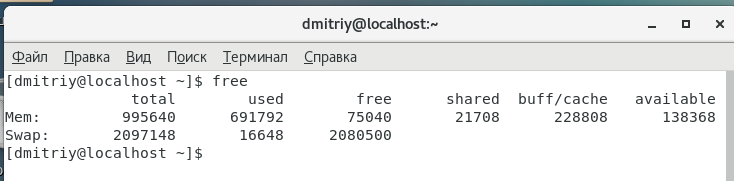
**Рисунок 22**. Файл */proc/version* и вывод команды *uname -a*

В файле */proc/meminfo* содержится информация о распределении и использовании оперативной памяти, а в файле */sys/devices/system/node/node0/meminfo* информация об узлах (в данном случае об узле под номером 0) (рисунок 23).



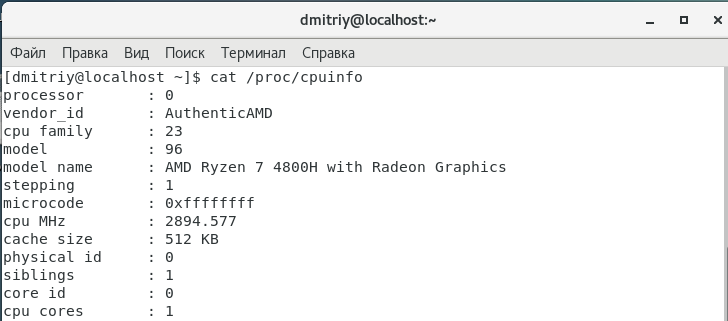
**Рисунок 23**. Файлы */proc/meminfo* и */sys/devices/system/node/node0/meminfo*

Команда *free* выводит тезисную информацию о доступном и используемом объеме оперативной памяти и раздела подкачки (рисунок 24).



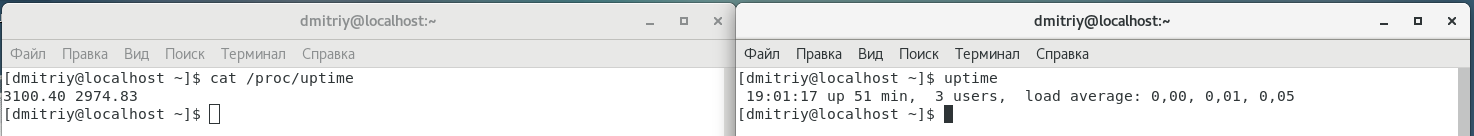
**Рисунок 24**. Вывод команды *free*

В распоряжение виртуальной машины предоставлено 1 ядро процессора, что подтверждается содержимым файла */proc/cpuinfo* (рисунок 25).



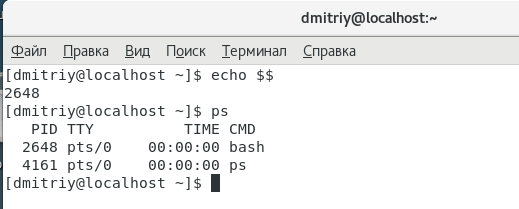
**Рисунок 25**. Информация о ЦП из файла */proc/cpuinfo*

В файле */proc/uptime* первое число – это время работы системы, второе число – это время, при котором ядра процессора не были нагружены. Команда *uptime* – вывод текущего времени, время работы системы, количество пользователей, и средняя нагрузка системы (рисунок 26).



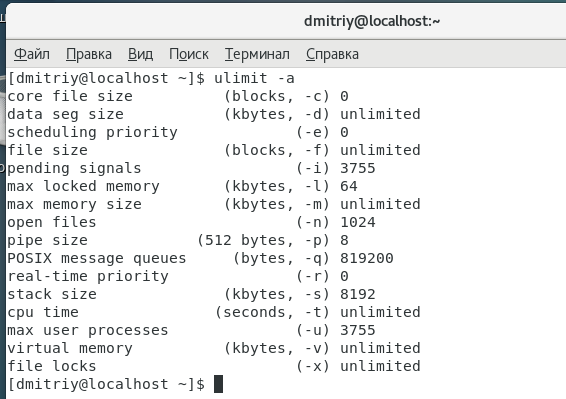
**Рисунок 26**. Информация о времени работы системы

В результате выполнения команды *echo $$* выводится PID процесса, в котором выполнялась эта команда. В данном случае, выведенный PID символьной ссылки совпадает с PID текущего сеанса оболочки bash (рисунок 27).



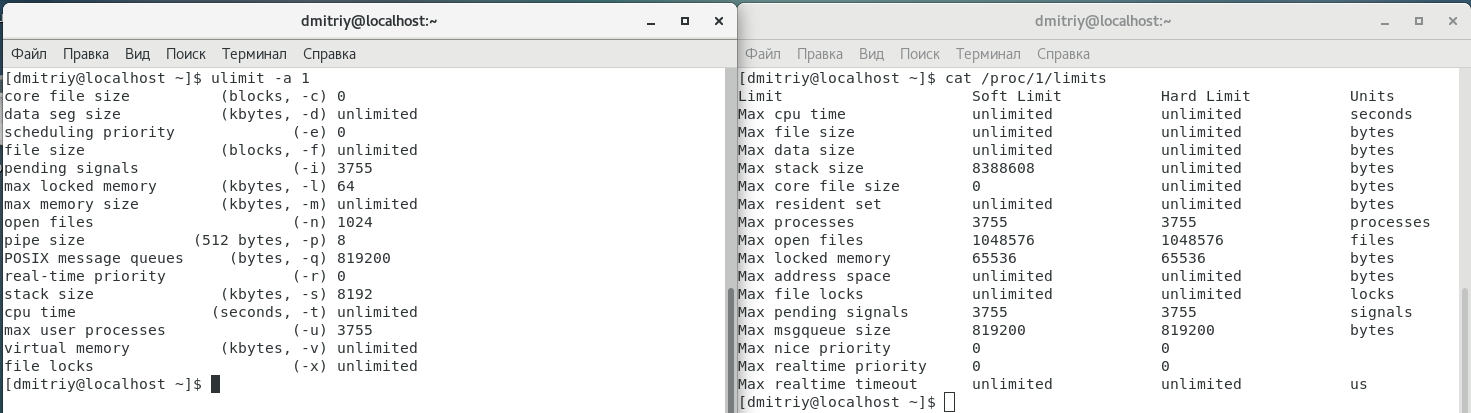
**Рисунок 27**. Изучение символьной ссылки */proc/self*

Команда *ulimit -a* позволяет вывести на экран пользовательские ресурсы оболочки (рисунок 28).



**Рисунок 28**. Пользовательские ресурсы оболочки

Также при вызове команды *ulimit -a 1* можно вывести на экран ресурсы процесса инициализации (init, с PID 1). Также данные о лимитах находятся в файле */proc/1/limits* (рисунок 29)



**Рисунок 29**. Ресурсы процесса инициализации

# **Вывод:**

В ходе выполнения лабораторной работы поставленные цели были успешно выполнены. Были освоены методы работы с процессами в ОС Linux, сигналы их завершения. Проанализированы системные файлы на соответствие вывода командам в терминале. Получены практические навыки по работе с заданиями в фоновом режиме и непосредственном запуске («на переднем плане»). Закреплены навыки работы с ОС Linux, с оболочкой ее командной строки.