МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Южно-Уральский государственный университет»

(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

Работа с процессами в Linux

ОТЧЕТ

по практической работе № 7

по дисциплине «Операционные системы семейства Unix/Linux»

Выполнил:

студент группы КЭ–301

/ Д.А. Иванов /

(подпись)

« » 2024 г.

Проверил:

преподаватель кафедры СП

/ П.Д. Ческидов /

(подпись)

« » 2024 г.

Оглавление

[1. команда ps 3](#_Toc165126506)

[2. Команда kill 4](#_Toc165126507)

[3. КОМАНДА killall 5](#_Toc165126508)

[4. утилита top 6](#_Toc165126509)

[5. команды nice и renice 8](#_Toc165126510)

[6. псевдофайловая система proc 9](#_Toc165126511)

# команда ps

Ход работы

Команда **ps** (сокращение от Process Status) – это утилита командной строки, с помощью которой можно посмотреть информацию о запущенных процессах в системе Linux.

Команда **ps** без аргументов выводит список запущенных процессов в текущем сеансе терминала. С параметром -**a** – все процессы, кроме фоновых. С параметром -**u** – процессы пользователя.

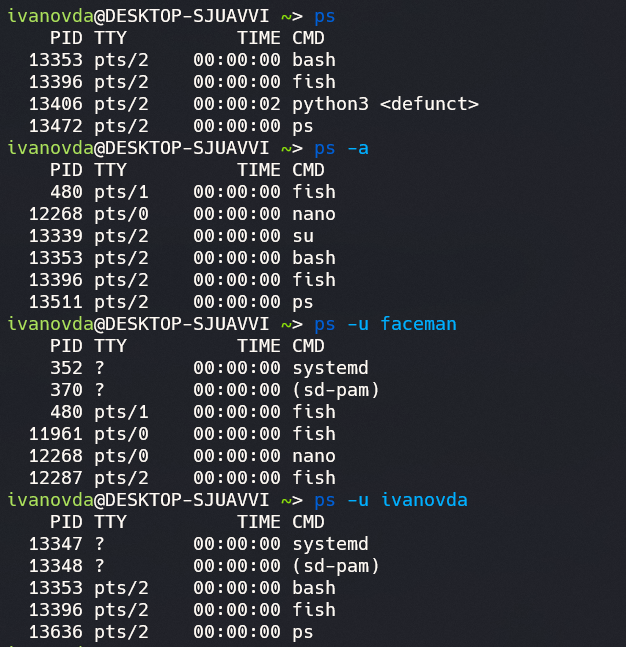


Рисунок 1 – Результат выполнения задания 1

# Команда kill

Ход работы

Команда **kill** это инструмент для управления процессами в операционных системах Linux. Она позволяет отправлять сигналы процессам, что позволяет контролировать их поведение. По умолчанию **kill** отправляет сигнал SIGTERM, который останавливает процесс.

В качестве примера запустим программу **nano** в одной сессии терминала и завершим её процесс из другой сессии с помощью команды **kill.** Для этого нужно узнать id процесса, с помощью команды **ps.**

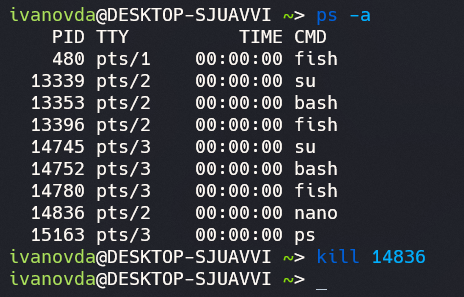


Рисунок 2 – Результат выполнения задания 2

После выполнения этой команды программа nano завершила свою работу.

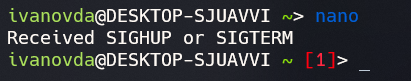


Рисунок 3 – Результат выполнения задания 2

# КОМАНДА killall

Ход работы

Команда **killall** используется для уничтожения любого запущенного процесса в системе на основе заданного имени.

Для примера также запустим программу **nano** и завершим её из другой сессии с помощью команды **killall.**



Рисунок 4 – Результат выполнения задания 3

# утилита top

Ход работы

Команда **top** – это утилита командной строки Linux, которая позволяет отслеживать запущенные процессы.

Окно можно условно разделить на две части. В верхней части находится информация о системе, общем использовании ресурсов процессора и памяти, раздела подкачки, и так далее. В нижней части окна расположен список запущенных процессов с информацией, отсортированных по определённому полю.

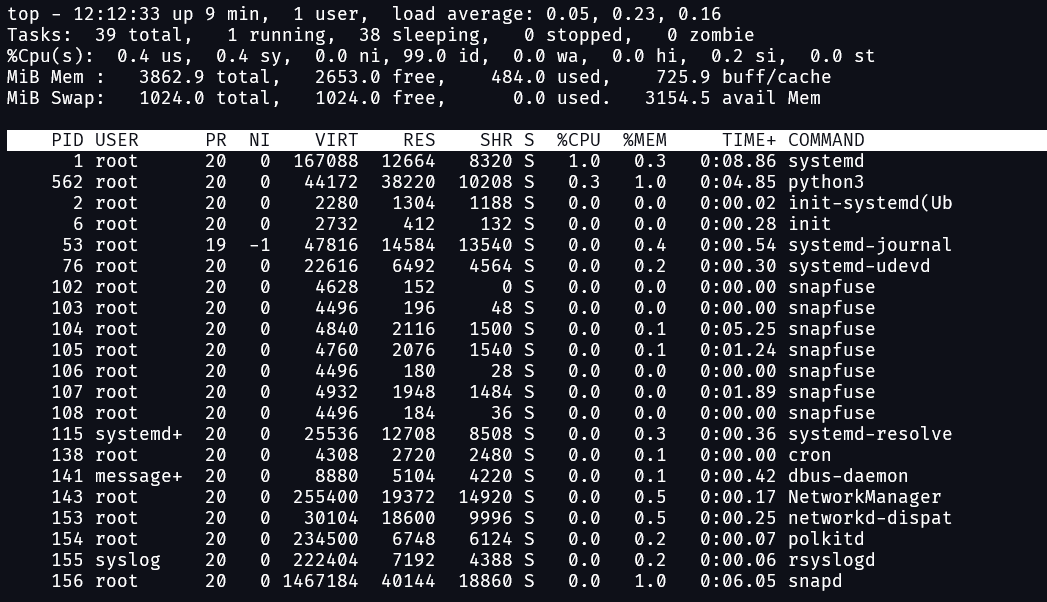


Рисунок 5 – Окно утилиты top

Утилита **htop** является более продвинутой и интерактивной альтернативой стандартной утилите top для мониторинга процессов в Linux-системах.

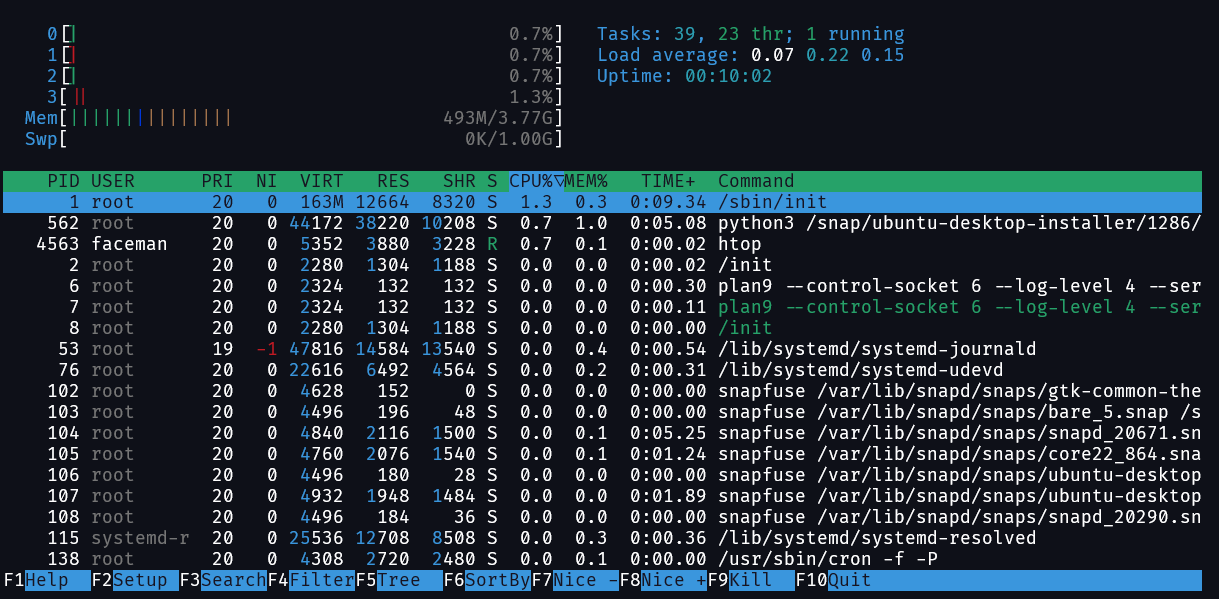


Рисунок 6 – Окно утилиты top

# команды nice и renice

Ход работы

Команды **nice** и **renice** в Linux предназначены для управления приоритетом выполнения процессов.

Команда **nice** позволяет запускать программы с измененным приоритетом для планировщика задач. Приоритет процесса измеряется числом в диапазоне от -20 до 19, где -20 - наивысший приоритет, а 19 - наименьший. По умолчанию назначается приоритет 0. Например, **nice -n** **15** **nano** запустит процесс nano с приоритетом 15.

Команда **renice** используется для изменения приоритета уже запущенного процесса. Она позволяет изменить приоритет процесса на лету, что может быть полезно для оптимизации использования ресурсов системы. Например, **renice 4 -p 100** установит приоритет процесса с PID 100 на 4.

Таким образом, команды **nice** и **renice** предоставляют возможность управлять приоритетами процессов в Linux, что позволяет оптимизировать использование ресурсов системы.

# псевдофайловая система proc

Ход работы

**/proc** – это виртуальная файловая система в Linux, которая предоставляет доступ к информации о системе и запущенных процессах.

Она позволяет получать информацию о ядре, процессорах, памяти, устройствах, файловых системах и многом другом без использования специальных утилит.

Примеры команд для вывода информации:

* Версия ядра: cat /proc/version;
* Информация о процессоре: cat /proc/cpuinfo;
* Использование оперативной памяти: cat /proc/meminfo;
* Список устройств: cat /proc/devices;

Таким образом, /proc предоставляет прямой доступ к внутренней информации ядра Linux, позволяя получать разнообразные сведения о системе без использования дополнительных команд или утилит.