|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **им. Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ: **Информатика и системы управления**

КАФЕДРА: **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ: **9.03.03 Прикладная информатика**

**Отчет**

**по лабораторной работе №4**

**Тема:** Исследование способов организации оперативной памяти и взаимодействия процессов.

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-54Б |  |  | Д.О. Кошенков |
|  | (группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**ВВЕДЕНИЕ**

**Цель лабораторной работы**

Цель данной работы — получение теоретических и практических сведений об управлении процессами, потоками и оперативной памятью в UNIX-подобных системах и в Linux в частности.

Прежде, чем читать файл с диска, ОС пытается найти его в кэше. Если какой-то файл недавно читали, второй раз он будет быстро прочитан из памяти. Этот вид кэша занимает больше всего места в памяти. Вы не можете явно указать сколько мегабайт может система использовать под кэш, но можно настроить скорость удаления просроченных страниц из кэша.

Размер страничного кэша указан в колонке cached

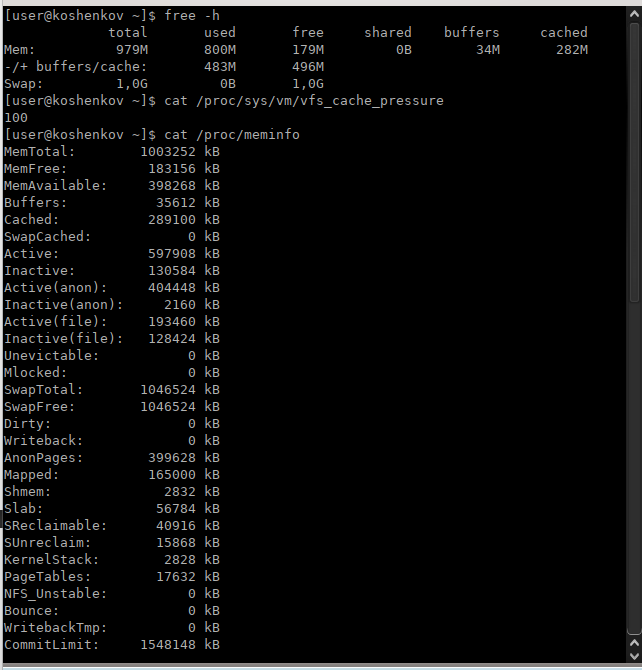


Рисунок 1 – Вывод информация о памяти.

Консольный интерфейс UNIX поддерживает многозадачность. Для того чтобы свернуть приложение, необходимо нажать Ctrl-Z, после чего на экран консоли будет выведено сообщение

1. **Открыть в текстовом браузере некую страницу и перевести его в фоновый режим**

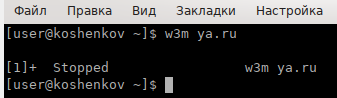
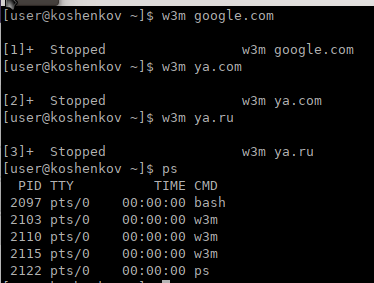


Рисунок 2 – запуск приложения в фоновом режиме.

1. **Запустить ещё два экземпляра текстового браузера в фоновом режиме**



Цифра в квадратных скобках обозначает номер задания, он необходим для некоторых команд

1. **Найти процесс, максимально нагружающий процессор**

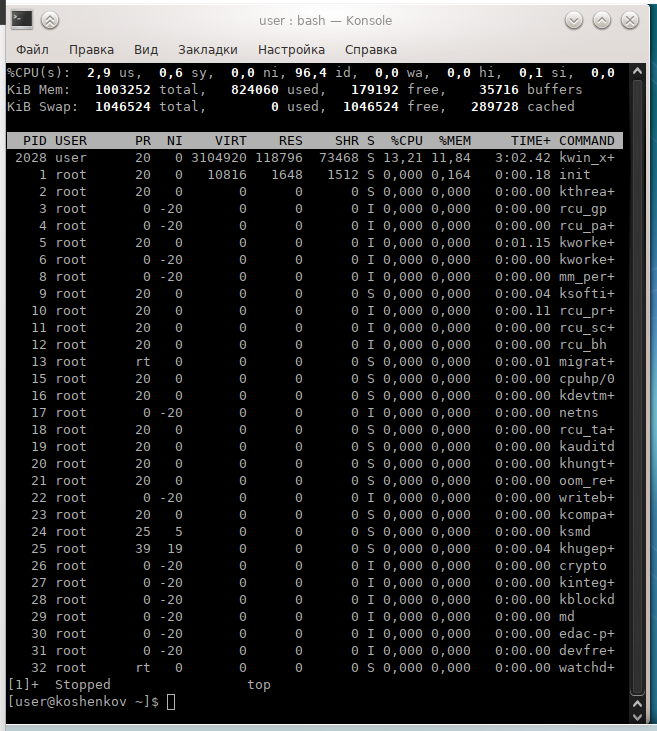


Рисунок 3– отображение списка процессов и потоков.

Как видно из рисунка ниже, больше всего процессор загружает процесс под номером 2028. Процент загрузки процессора этим процессом 13.21.

1. **Вывести список процессов текущего пользователя**

С помощью команды top –u user я вывел только те процессы, которые принадлежат текущему пользователю (user) =

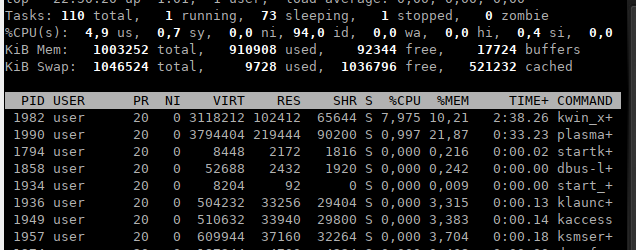


Рисунок 5 – вывод процессов принадлежащих определенному пользователю

1. **“Убить” первый процесс браузера, в котором открыта 1 страница**



Рисунок 6 – принудительное завершение процесса.

С помощью команды kill -9 <PID> я принудительно завершил процесс w3m [ya.ru](http://www.yandex.ru/), который до этого находился в фоновом режиме

1. **Вывести список всех процессов всех пользователей. Это делается с помощью команды ps axu**

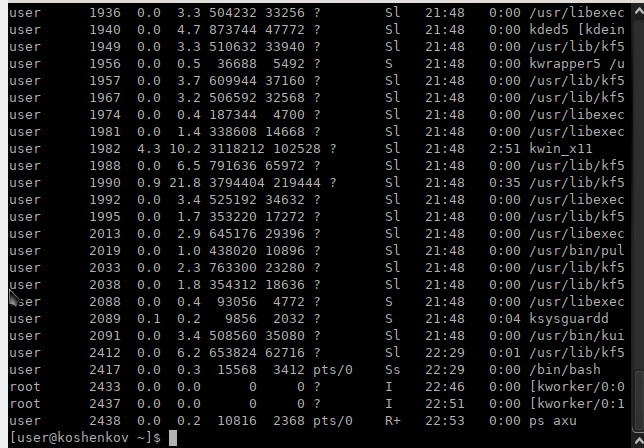


Рисунок 7 – вывод всех процессов всех пользователей

1. **Просмотреть список процессов постранично**

Постранично можно посмотреть с помощью команды ps axu | more. Она запускает два процесса в конвейере. Выход команды ps axu направляется на вход команды more, которая осуществляет требуемый вывод по страницам.

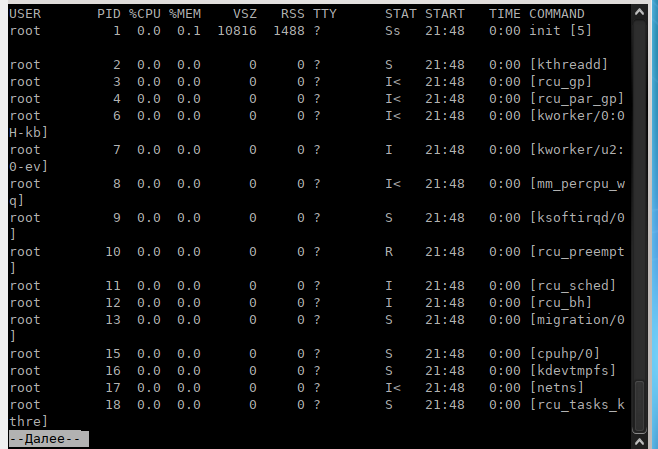
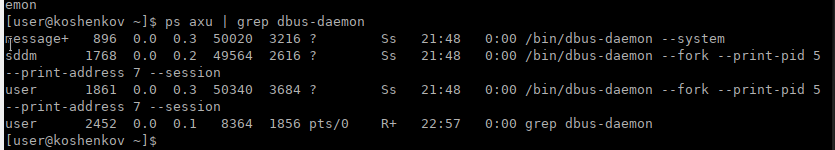


Рисунок 8 – вывод всех процессов постранично.

1. **Отобрать из вывода команды ps строку, соответствующую процессу «dbus-daemon», определить, где лежит её выполняемый файл и с какими параметрами он запущен.**

Сначала выполним команду ps axu | grep dbus-daemon, чтобы поулчить всю необходимую информацию. Grep отберет из входных данных только те строки, которые содержат первый параметр (dbus-daemon)



1. **Записать в файл с именем, содержащим текущее время, строку «-------» и список процессов**

Чтобы записать текст в файл нужна команда echo ”что надо записать”>>файл, куда надо записать. Также применим здесь операцию последовательного выполнения команд &&. В итоге получим

(echo “------“ && ps axu) >> "$(date).txt", результат на рисунке ниже

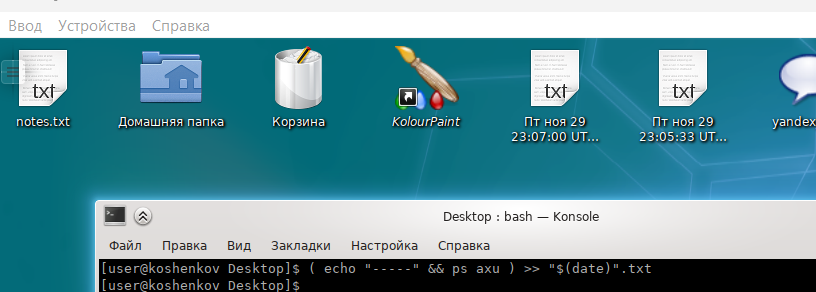


Рисунок 9 – создание конвейера команд

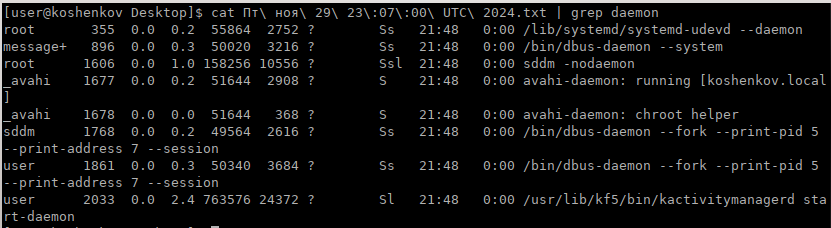
1. **Выполнить команду в фоновом режиме с отсрочкой запуска на 1 минуту. Продемонстрировать, что команды выполнилась именно через минуту.**

Для демонстрации лучше всего использовать команду date, которая выводит текущую дату и время. Для отсрочки запуска используется команда sleep <секунды>



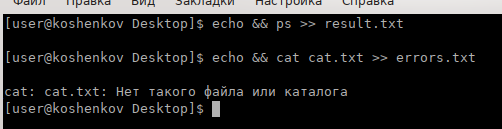
1. **Отобрать из одного из сформированных файлов строки, относящиеся к одному из процессов.**

Буду искать все записи о демонах в файле, созданном ранее с помощью команды grep.



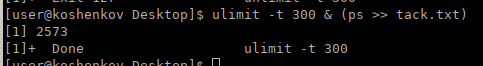
1. **Вывести результаты работы произвольной команды в один файл, а сообщения об ошибках в другой. Продемонстрировать правильность работы.**

Запишем в файл result.txt результат работы команды ps, а в файл error.txt результат команды cat cat.txt. Команда cat cat.txt является ошибочной, т.к. в нет такого файла/

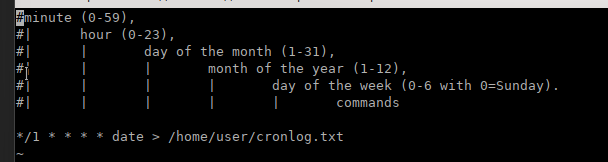


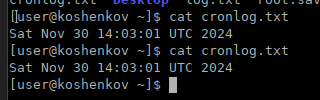
1. **Выполнить произвольную команду с ограничением использования процессорного времени 300 секунд и выводом результатов и сообщений об ошибках в один файл.**

Для ограничения использования процессорного времени используем команду ulimit –t (ограничение по времени). Выполним команду ulimit –t 300 & (ps >> task.txt)



1. **Настроить cron на выполнение команды ежедневно в заданное время. Продемонстрировать правильность работы.**





**Выводы.**

Linux имеет гибкие механизмы управления процессами и потоками. Процессы могут посылать друг другу сигналы или сообщения, обмениваться данными через файлы или каналы. Суперпользователь может изменять приоритеты процессов. Также Linux имеет стандартные гибкие средства мониторинга ресурсов