***Факультет инфокоммуникационных систем и технологий***

**КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ**

**Отчёт по практической работе №7**

**по дисциплине «Операционные системы, среды и оболочки»**

**Выполнил:** *студент гр. ИСТ 22/1 Ветров Е.М.*

**Преподаватель:** *доцент каф ИТУС* Исаева Г.Н.

***Цель работы:*** Закрепить представление об основных возможностях подсистемы управления процессами ядра ОС Linux Ubuntu.

1. Исследуйте команду ps: приведите её ключи и проверьте их действие. Приведите расшифровку столбцов таблицы, выдаваемой командой ps -p .

Команда ps используется для вывода информации о текущих процессах в системе. Некоторые ключи команды:

- -a: показывает процессы всех пользователей

- -u: выводит расширенную информацию о процессах

- -x: выводит процессы, не имеющие контрольных терминалов

- -p: выводит информацию о процессах с определёнными идентификаторами

- -e: показывает информацию обо всех процессах, запущенных на системе

При использовании команды ps -p <список идентификаторов процессов>, где <список идентификаторов процессов> - это список номеров PID процессов разделенный запятыми, выводится таблица со следующими столбцами:

- PID - идентификатор процесса

- TTY - управляющий терминал (если есть)

- TIME - время использования процессора

- CMD - команда, запустившая процесс

На рисунке 1 и 2 мы можем наблюдать команды ps -\*

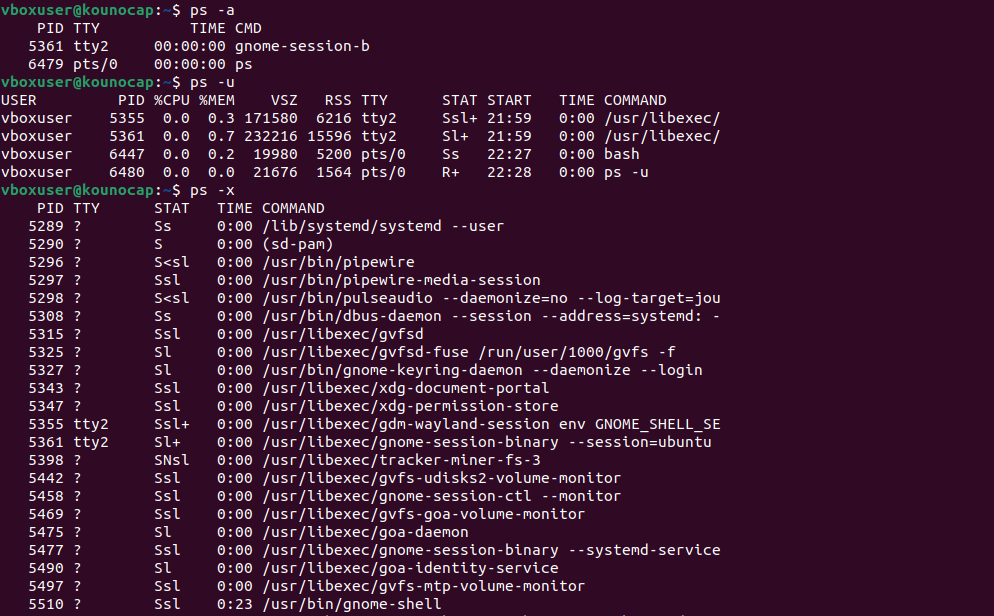


Рисунок 1. Результат команды ps -\*

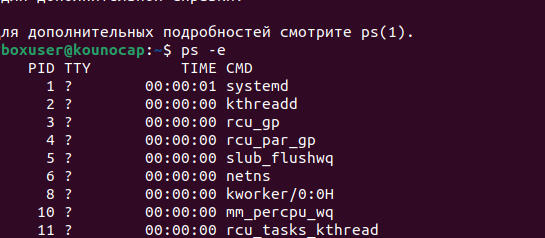


Рисунок 2 Результат команды ps -\*

2. Выполните команду вывода приветствия «Hello World!» в 11:00 часов в каждые 6,7,8,9 дни января, февраля и марта, а приветствие «Good day!» - каждый четный час каждого понедельника.

Для выполнения такой команды в Linux можно использовать утилиту cron. Ниже приведен пример команды, которая выводит приветствие "Hello World!" в 11:00 каждые 6,7,8,9 дни января, февраля и марта и приветствие "Good day!" - каждый четный час каждого понедельника:

```

0 11 6-9 1-3 \* echo "Hello World!"

0 \*/2 \* \* 1 echo "Good day!"

```

Эта команда добавляет задания в cron для выполнения указанных действий по расписанию. Для ее выполнения необходимо открыть терминал и ввести команду `crontab -e`. Затем необходимо вставить указанные выше строки в файл cron и сохранить его. Команда будет выполняться автоматически в заданные дни и время.

На рисунке 3 показано открытие команды и терминала через crontab-e и запись кода.

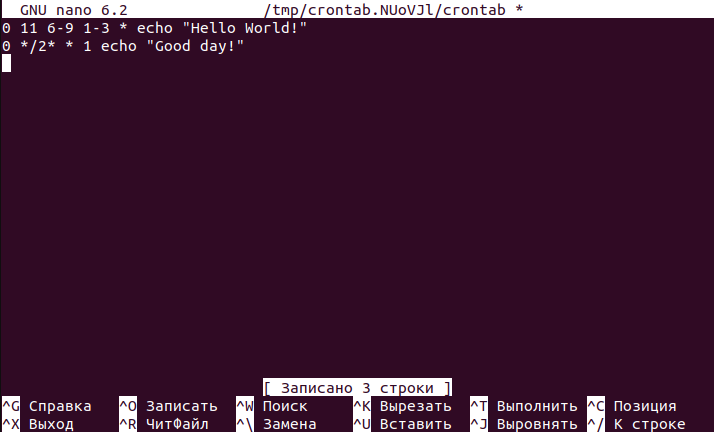


Рисунок 3. Открытие термина и ввод команды для ее выполнения

3. Приведите команду ps таким образом, чтобы можно было посмотреть подробную информацию обо всех процессах, запущенных от имени текущего пользователя; список процессов должен быть упорядочен по степени использования процессорного времени.

Команда для просмотра подробной информации обо всех процессах, запущенных от имени текущего пользователя, упорядоченных по степени использования процессорного времени:

```

ps -U $(whoami) -u $(whoami) -o pid,%cpu,%mem,comm | sort -nrk 2

```

Описание ключей команды:

- -U: выбрать процессы только для указанного пользователя

- -u: выводить процессы только для указанных пользователей

- -o: определить свой вывод формата, в данном случае - идентификатор процесса, % использования процессора, % использования памяти и команда процесса

- | sort -nrk 2: отсортировать список по столбцу %CPU в обратном порядке

Столбцы таблицы:

- PID: идентификатор процесса

- %CPU: процент использования процессора

- %MEM: процент использования памяти

- COMM: команда запущенного процесса

На рисунке 4 показан команда для просмотра информации обо всех процессах

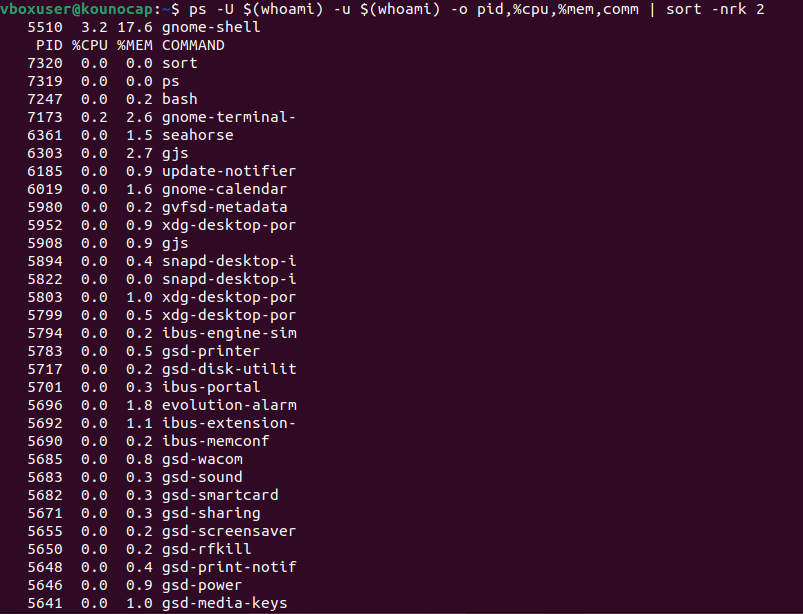


Рисунок 4 Команда для просмотра подробной информации обо всех процессах

4. Приведите пример использования данных команд: nice [-n ] renice -n kill –s

1. Команда "nice"

ssh username@ip\_address

nice -n 10 tar -czvf backup.tar.gz /home/user

Эта команда подключится к удаленному серверу по SSH (замените "username" и "ip\_address" на соответствующие значения), а затем создаст архив с использованием команды "tar" с приоритетом выполнения 10.

2. Команда "renice"

ssh username@ip\_address

renice -n 10 1234 5678

Эта команда изменит приоритет выполнения процессов с идентификаторами 1234 и 5678 на 10.

3. Команда "kill"

ssh username@ip\_address

kill -s TERM 1234

Эта команда отправит сигнал "TERM" процессу с идентификатором 1234, что приведет к остановке этого процесса на удаленном сервере.

На рисунке 5 показано команды nice, renice, kill.

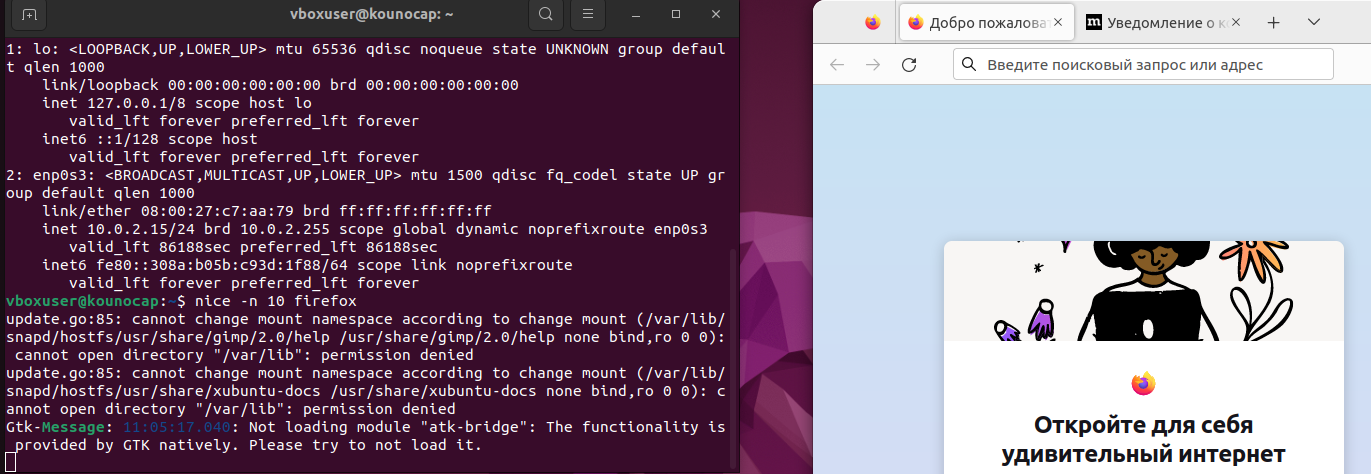


Рисунок 5 Приведения команд nice renice kill

5. Организуйте взаимодействие двух процессов посредством передачи данных через файл. Для этого в разных консолях запустите каждый из скриптов. Тогда, переключаясь между консолями можно вести управление и наблюдение за их работой. Тексты скриптов приведены в данном пункте ниже. Короткий скрипт считывает информацию с консоли. Второй скрипт, более длинный, выводит её на экран, причем таким образом, чтобы считывание первым скриптом строки «QUIT» приводило к завершению работы обработчика. #!/bin/bash while true; do read LINE echo $LINE >> data.txt done #!/bin/bash (tail -n 0 -f data.txt) | while true; do read LINE; case $LINE in QUIT) echo "exit" killall tail exit ;; \*) echo $LINE ;; esac done

Из предложенных скриптов можно организовать взаимодействие двух процессов через файл data.txt, находящийся в одной директории с файлами скриптов.

1. Создадим и запустим первый скрипт:

```

#!/bin/bash

while true; do

read LINE

echo $LINE >> data.txt

done

```

Этот скрипт будет записывать данные, введенные пользователем, в файл data.txt.

2. Создадим и запустим во второй консоли второй скрипт:

```

#!/bin/bash

(tail -n 0 -f data.txt) | while true; do

read LINE;

case $LINE in

QUIT)

echo "exit"

killall tail

exit

;;

\*)

echo $LINE

;;

esac

done

```

В этом скрипте мы выполняем чтение файла data.txt и выводим содержимое на экран. Если в файле появляется строка «QUIT», то происходит завершение работы обработчика.

3. Для взаимодействия между скриптами необходимо переключаться между консолями и вводить текст в первый скрипт. Второй скрипт будет автоматически выводить на экран содержимое файла data.txt.

Пример взаимодействия между скриптами:

- в первом окне консоли запустим первый скрипт: `./script1.sh`

- во втором окне консоли запустим второй скрипт: `./script2.sh`

- в первом окне консоли вводим текст для передачи во второй скрипт

- во втором окне выводится содержимое файла data.txt (введенный текст)

- для завершения работы обработчика во втором окне необходимо ввести строку "QUIT" в первом окне

На рисунке 6 показано чтение и ввод текста через файл data.txt и вывод на экран.

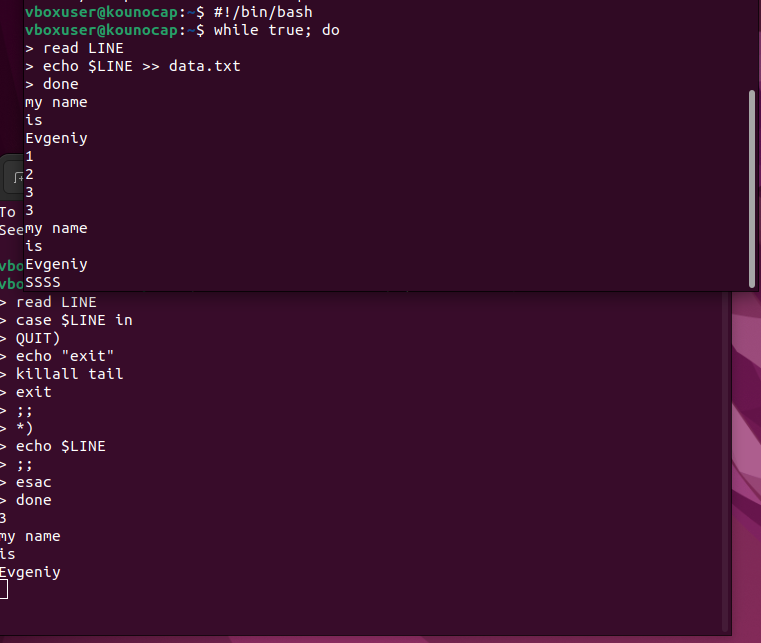


Рисунок 6 Чтение и ввод текста через файл

6. Составьте конвейер команд, где происходит активация какого либо процесса, затем по таблице идентификаторов процессов определите его PID и завершите этот процесс, например, таким образом: kill –s SIGSTOP 123. Проанализируйте действия и результаты по командам.

Допустим, у нас есть процесс "my\_process", который мы хотим приостановить, а затем завершить. Мы можем использовать конвейер команд для этого. Вначале мы запустим процесс "my\_process", а затем используем команды для получения его идентификатора процесса и завершения его:

```

firefox & pid=$! && sleep 5 && kill -s SIGSTOP $pid && sleep 5 && kill -s SIGKILL $pid

```

Эта команда запустит процесс "firefox" в фоне и запишет его идентификатор процесса в переменную "pid". Затем мы остановим процесс на 5 секунд с помощью команды "kill -s SIGSTOP" и снова подождем 5 секунд. Наконец, мы прервем выполнение процесса с помощью команды "kill -s SIGKILL".

Результаты будут зависеть от конкретного процесса "my\_process". Если процесс успешно останавливается и завершается, то мы увидим вывод в консоль, подтверждающий это. Если же процесс не реагирует на сигналы, то мы можем попробовать отправить сигнал "SIGTERM" или "SIGINT" вместо "SIGKILL", чтобы попытаться корректно завершить процесс.

На рисунке 7 результат команды firefox & pid=$! && sleep 5 && kill -s SIGSTOP $pid && sleep 5 && kill -s SIGKILL $pid

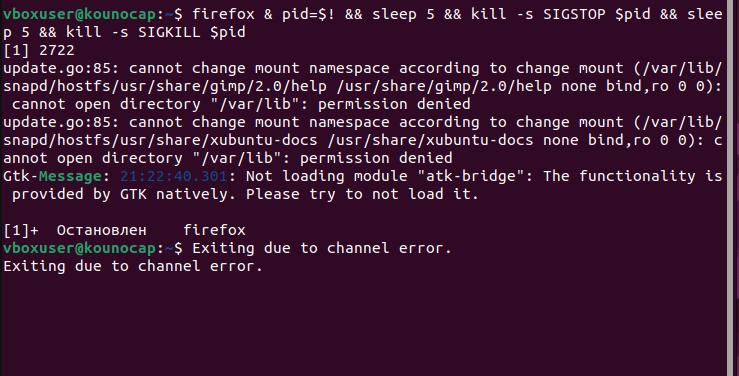


Рисунок 7 Резултат команды firefox & pid=$! && sleep 5 && kill -s SIGSTOP $pid && sleep 5 && kill -s SIGKILL $pid