Лабораторная работа №3 Доскоч Роман. Вариант 5.

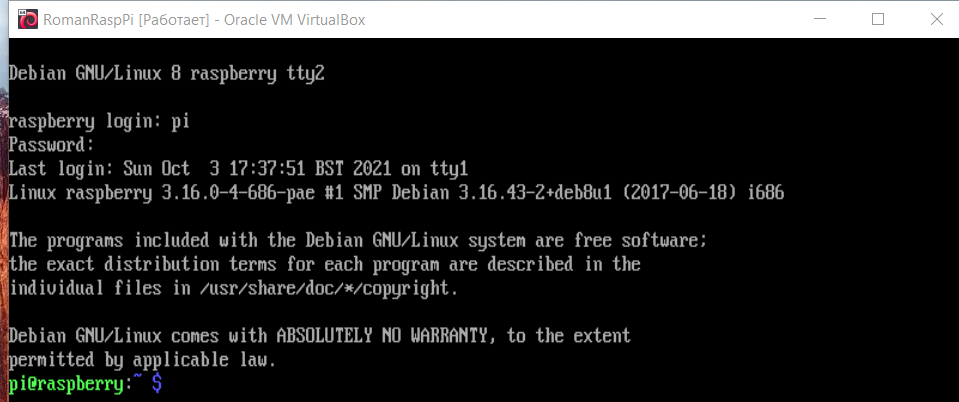
**Цель работы.**

Цель работы – изучение файловой системы ОС Unix/Linux и основных функций для работы с каталогами и файлами; исследование методов создания процессов, основных функций создания и управления процессами, обмена данными между процессами.

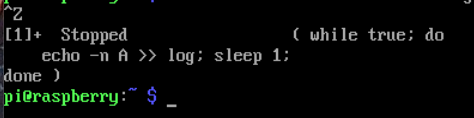
**Задание 3.1**

Вошел в tty2 ctrl + alt + F[1..6]

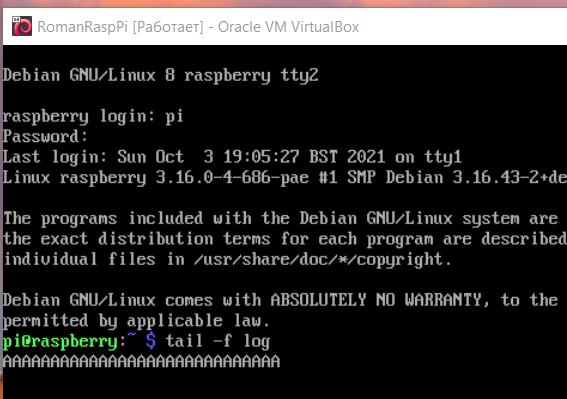
1. Используйте учетную запись, созданную в одной из предыдущих лабораторных работ. Войдите в систему на виртуальных терминалах 1 и 2 (tty1, tty2) под вашей учетной записью.

****

2. Переключитесь в терминал tty1 и запустить процесс, выполняющий следующие команды: $ (while true; do echo -n A >> log; sleep 1; done)

****

3. Заметьте, что сейчас этот терминал занят исполнением запущенного процесса, который исполняется на переднем плане. Этот процесс присоединяет символ "А"к файлу ~/log через каждую секунду. Чтобы визуально проверить это, переключитесь в виртуальный терминал tty2 и выполните следующую команду: $ tail -f log Вы должны увидеть последовательность символов, длина которой возрастает.

****

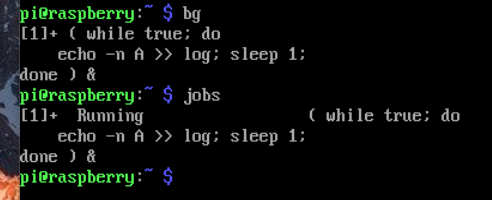
4. Переключитесь в виртуальный терминал tty1 и приостановите работающий процесс, нажав клавиши <ctrl+Z>. Командная оболочка сообщит, что процесс остановлен и выдаст вам номер задания [1]. Переключитесь в виртуальный терминал tty2 и визуально проверьте, что файл ~/log больше не увеличивается.

5. Переключитесь в виртуальный терминал tty1 и возобновите работу процесса в фоновом режиме. Используйте команду jobs, чтобы проверить, что задание [1] снова работает.

$ bg

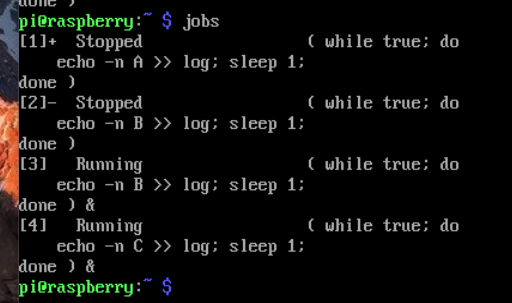
$ jobs

Переключитесь в виртуальный терминал tty2 и визуально проверьте, что файл ~/log снова увеличивается.

****

6. Переключитесь в виртуальный терминал tty1 и запустите еще два процесса, выполнив следующие команды: $ (while true; do echo -n B >> log; sleep 1; done) & $ ^B^C Вторая команда просто запускает предыдущую команду, заменяя символ "B"символом "C".

7. Выполните команду jobs и проверьте, что все три процесса работают. Переключитесь в виртуальный терминал tty2 и визуально проверьте, что файл ~/log снова увеличивается путем добавления символов "A" "B"и "C"через каждую секунду.

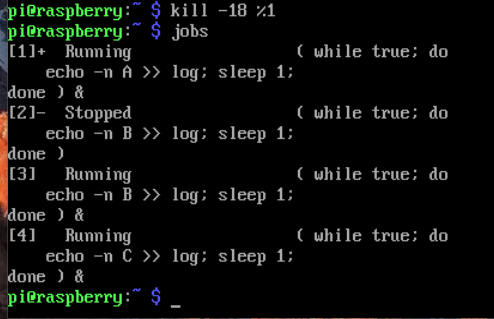
****



8. В пункте 4 вы приостановили исполнение процесса переднего плана путем нажатия клавиш . В действительности эта комбинация нажатых клавиш посылает процессу сигнал. Используйте команду kill, чтобы получить список сигналов и соответствующие им имена и номера. Затем выполните команду kill, послав сигнал SIGSTOP заданию [1], чтобы приостановить его работу. Переключитесь в виртуальный терминал tty1 и выполните следующие команды: $ kill -l $ kill -19 %1

9. Выполните команду jobs и проверьте, что задание [1] остановлено. Переключитесь в виртуальный терминал tty2 и визуально проверьте, что задание [1] остановлено.

10. Возобновите выполнение задания [1], используя команду kill, которая посылает процессу сигнал SIGCONT (18). Используйте команду jobs и виртуальный терминал tty2 для проверки того, что все три задания опять работают.



11. Завершите работу всех трех процессов. Если вы не задаете сигнал, который нужно послать процессу, то команда kill посылает по умолчанию сигнал SIGTERM (15), который вызывает завершение процесса. После посылки сигналов заданиям [2] и [3], используйте команду jobs, чтобы проверить завершение работы этих заданий: $ kill %2 %3 $ jobs

12. Чтобы завершить работу последнего процесса, выполните команды: $ fg $

13. Выполните команду jobs и проверьте, что больше заданий не выполняется. Переключитесь в виртуальный терминал tty2 и визуально проверьте, что файл ~/log не увеличивается. Остановите исполнение команды tail, нажав клавиши , и завершите сеанс на виртуальном терминале tty2.

****

14. Переключитесь в виртуальный терминал tty1 и удалите файл ~/log.

**rm log**

Задание 3.2

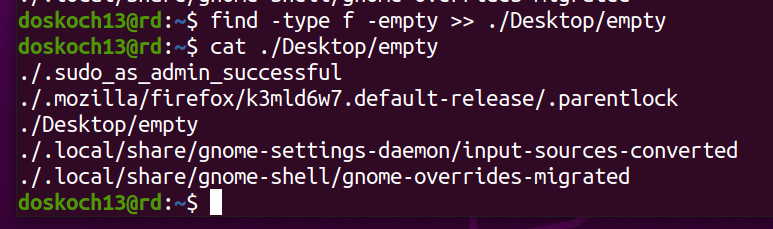
Отличие от Задание 3.1 в изменении приоритета потока

renice [priority] -p [pid]

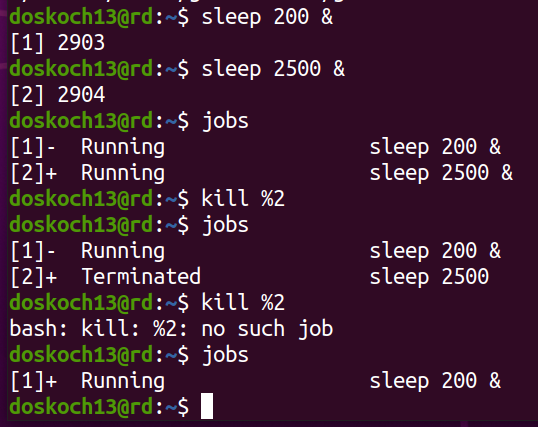
Задание 3.3

Управление процессами и планирование заданий Изучите примеры

1. Найдите файлы а) пустые б) скрытые в домашнем каталоге в фоновом режиме и результат сохраните в файл со своей фамилией.



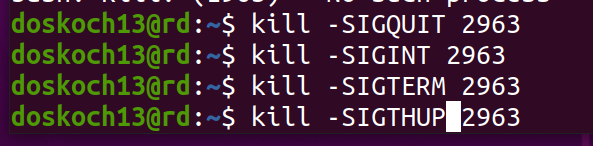
2. Запустите в фоновом режиме два задания: sleep 200 и sleep 2500, выведите информацию о состоянии заданий. Снимите с выполнения второе задание, выведите информацию о заданиях.



3. Выполните команду exec ls -R /etc. Изучите её поведение.

Вылетел терминал…

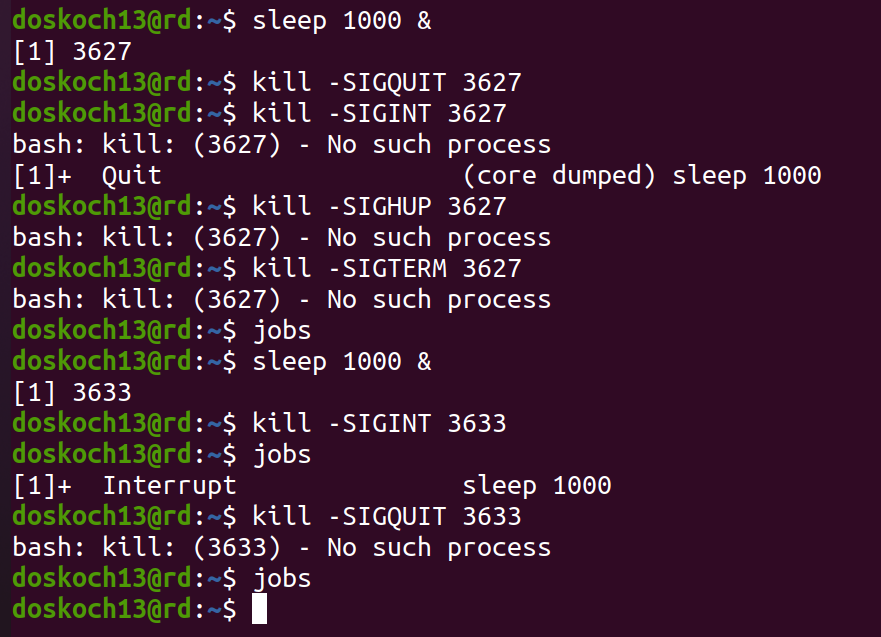
4. Запустите порожденную оболочку bash. Исследуйте, посылая родительской оболочке сигналы TERM, INT, QUIT и HUP, что при этом происходит?



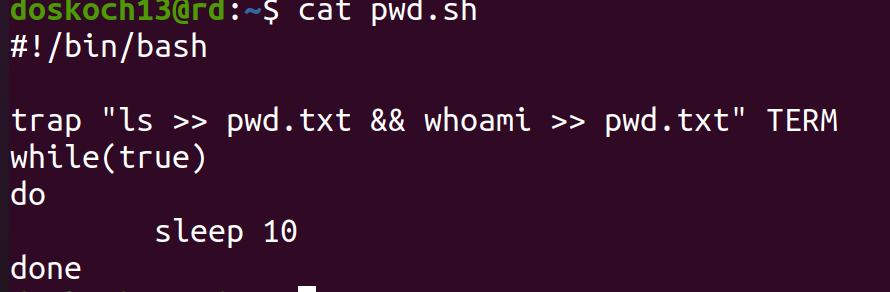
5. От имени обычного пользователя пошлите сигнал KILL любому процессу, запущенному от имени другого пользователя. Что произойдет?

Завершение процесса.

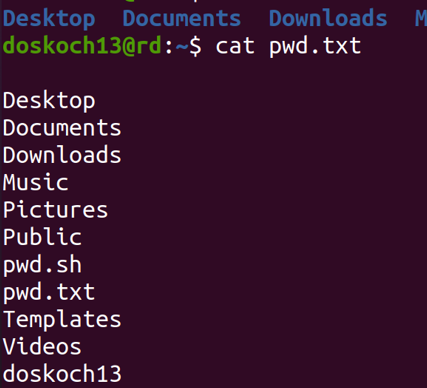
6. Запустите в фоновом режиме команду sleep 1000. Проверьте, на какие сигналы из следующих: TERM, INT, QUIT и HUP, реагирует эта команда.



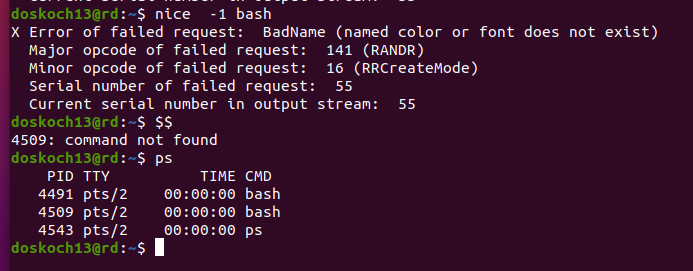
7. Запрограммируйте оболочку так, чтобы при получении ей сигнала TERM создавался файл pwd.txt, содержащий информацию о текущем каталоге и текущем пользователе.



8. Запустите порожденную оболочку. Работает ли в ней созданный обработчик?



9. От имени обычного пользователя попытайтесь запустить оболочку bash со значением nice number, равным 1. Какое сообщение выводится?



11. При помощи команды at сделать так, чтобы ровно через 5 минут от текущего времени произошла запись списка всех процессов в файл с именем, содержащим в своём названии системное время на момент записи.

*at now + 5 minute*

*ps >> ps.txt*

12.При помощи команды at организовать обычное завершение работы браузера firefox или chrome в 16:00.

*. at 4pm*

*killall -9 firefox-bin*

13.Сделать при помощи cron так, чтобы команда updatedb запускалась раз в сутки, каждый час, каждые 5 минут.

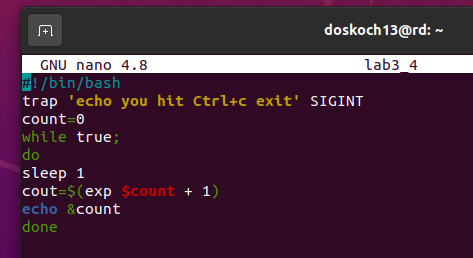
*0 0 \* \* \* updatedb*

*0 \* \* \* \* updatedb - каждый час*

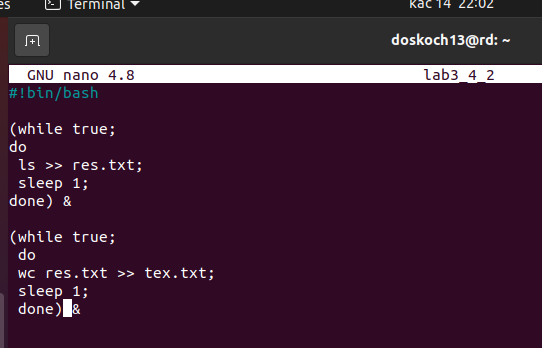
*\*/5 \* \* \* updatedb - каждые 5 минут*

Задание 4

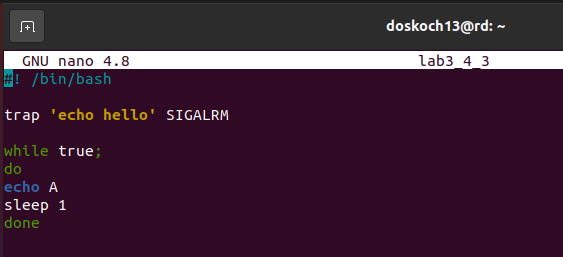
1)



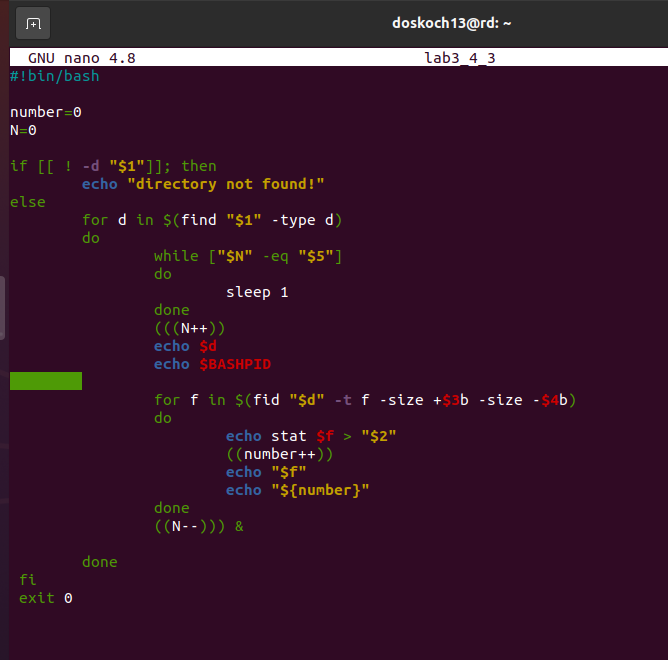
2)



3)



Задение 5 Вариант 4



Контрольные вопросы

1) *Объясните понятия процесса и ресурса. Какое их значение в организации вычислительного процесса в ОС Linux?*

Процесс – набор выполняемых заданий, инструкций. Ресурс – это то, что выделяется процессу для выполнения данных задач. Процесс представлен в памяти в виде дескриптора процесса, который включает: исполняемый текст программы и управляющая информация. Концепция процессов обеспечивает свойство многозадачности ОС, т.е. когда в активном состоянии могут находиться несколько задач, между которыми разделяется процессорное время.

2) *Какая информация содержится в описателях процессов? Как просмотреть их содержание в процессе работы с системой?*

Описатель процесса – это дескриптор. Дескриптор включает в себя исполняемый текст программы задачи, управляющую информацию (контекст процесса), необходимую для выделения и использования выделенных процессу ресурсов: приоритет, необходимое адресное пространство, вектор прерывания (слово состояния программы), величину кванта процессорного времени, признаки состояния процесса и пр.

Посмотреть: ps -l, ps.

3) *Какими способами можно организовать выполнение программ в фоновом режиме?*

Для запуска фонового (параллельного с другими потомками) процесса в командную строку необходимо и достаточно последним символом добавить знак &

4) *Какие особенности выполнения программ в фоновом режиме? Как избежать вывода фоновых сообщений на экран и прерывания выполнения фоновых программ при прекращении сеанса работы с системой?*

Фоновые процессы обладают некоторыми недостатками:

- не допускают ввода с клавиатуры;

- обеспечивают вывод на экран, но при этом нарушают целостность вывода диалогового процесса.

Общепринятый прием исключения влияния фонового вывода на интерактивную работу:

$<командная\_строка> > имя\_файла.out &

- <командная\_строка> планирует задание для фонового режима;

- перенаправляет вывод вместо экрана в указанный файл головного каталога пользователя.

особенности работы с фоновым режимом:

- выполняемая в фоновом режиме программа (команда), требующая стандартного ввода, должна читать его из файла с использованием перенаправленного ввода;

- программа, выполняемая в фоновом режиме, не может быть прервана <Ctrl\*C>, так как она отсоединяется от клавиатуры и может быть прекращена только с помощью команды kill или выходом из системы;

- выход из системы exit надо выполнять два раза: для завершения фонового процесса и завершения основного процесса shell.

5) *Как пользователь может повлиять на распределение ресурсов между активными процессами?*

С помощью команды nice.

Если при выполнении задания образуются несколько порожденных процессов, то все они имеют одинаковый приоритет равный родительскому. В этом случае все процессы получают ресурсы равными долями (простой режим разделения времени). При необходимости выделения наиболее важных родительских процессов порожденным второстепенным можно понизить приоритет с помощью команды.

6) *Как можно прервать выполнение активных процессов? Какая информация для этого необходима и откуда она извлекается?*

Завершение процессов – одна из функций управления процессами. Прекратить выполнение любого процесса можно с помощью команды:

kill [-опции] PID1 [PID2......] — передает сигнал процессу PID.

7) *Перечислите базовые средства взаимодействия процессов в Linux.*

Каналы, сигналы, разделяемая память, очереди сообщений, сокеты

8) *Поясните особенности работы с каналами в Linux.*

Канал - это средство связи стандартного вывода одного процесса со стандартным вводом другого. Эта особенность широко используется даже в командной строке UNIX (в shell-е). Когда процесс создает канал, ядро устанавливает два файловых дескриптора для пользования этим каналом. Один такой дескриптор используется, чтобы открыть путь ввода в канал (запись), в то время как другой применяется для получения данных из канала (чтение)

9) *Почему отложенные вызовы не обрабатываются непосредственно обработчиком прерывания таймера?*

Потому что функции отложенных вызовов выполняются в системном контексте, а не в контексте прерывания