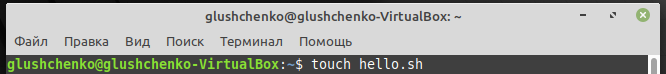
## Ваш первый сценарий bash

### Задания для выполнения

В данной работе мы напишем для вывода фразы "Hello world!" специальный сценарий.

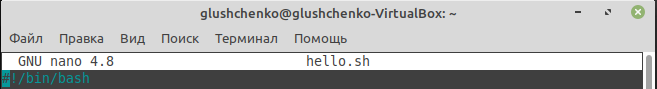
1. Создайте файл hello.sh.



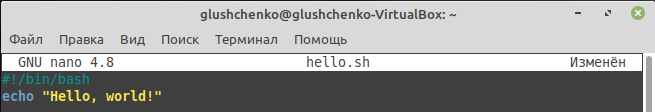
1. Откройте данный файл для редактирования в текстовом редакторе.



1. Запишите первую строку файла: #!/bin/bash. Это так называемый "шебанг" - специальная инструкция, сообщающая операционной системе, что данный файл нужно воспринимать именно как Bash-сценарий и использовать для его выполнения командную оболочку Bash.



1. Запишите вторую строку в файл: echo "Hello world!". Данная команда выводит на экран фразу "Hello world!".



1. Выполните полученный сценарий: bash hello.sh



1. Для того чтобы сценарий можно было выполнить как обычную программу, нужно сделать файл сценария исполняемым: chmod +x hello.sh



Чтобы показать, что файл может выполняться, введем команду ls -l, чтобы показать права доступа и группы.



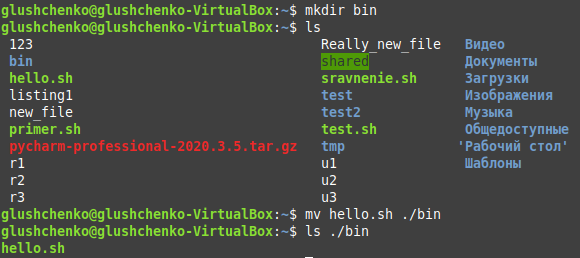
1. Попробуйте запустить сценарий как обычную программу: ./hello.sh.



1. Попробуйте запустить сценарий без указания пути: hello.sh. Данная команда не работает, так как она отсутствует в перечне путей в переменной $PATH. $PATH содержит список каталогов, разделенных “:”, в которых командная оболочка последовательно ищет исполняемый файл.



1. Создайте в домашнем каталоге каталог bin и переместите туда файл hello.sh.



1. Добавьте директорию ~/bin в переменную $PATH оператором конкатенации строк bash.



Проверяем:



1. Попробуйте запустить сценарий без указания пути: hello.sh. Так как сценарий теперь расположен по пути, который есть в $PATH, команда должна сработать и вывести на экран "Hello world!".

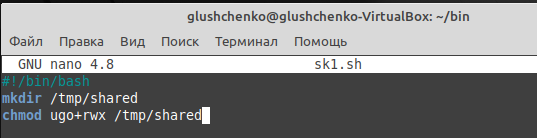


### Дополнительные задания

1. Разработать сценарий, который создаёт каталог /tmp/shared и устанавливает на него права доступа rwxrwxrwx.

Пишем свой скрипт сразу в ~/bin

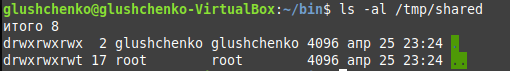




Где mkdir создает каталог, а chmod задает права доступа для всех групп



Проверим выполнение скрипта командой ls -al



1. Получите и выведите на экран, используя переменные окружения, имена текущего каталога и домашнего каталога.

Переменные окружения в Linux - это специальные переменные, определенные оболочкой и используемые программами во время выполнения. Они могут определяться системой и пользователем. Системные переменные окружения Linux определяются системой и используются программами системного уровня.

Например, команда PWD использует системную переменную, чтобы сохранять прежнюю рабочую директорию. HOME – переменная, отображающая домашний каталог пользователя. С помощью переменной **printenv**, которая выводит переменные среды выведем необходимые данные.



1. Создайте новую переменную NEWVAR со значением 1982.

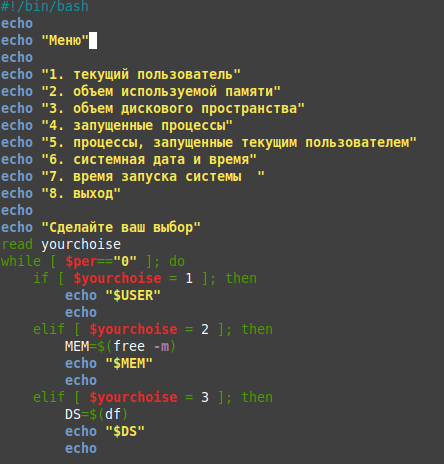


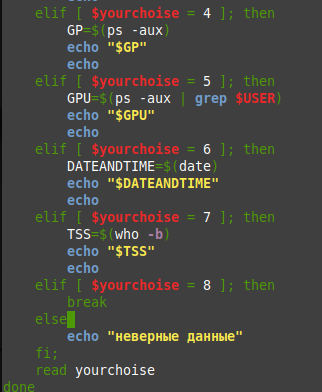
1. (\*) Создать скрипт, осуществляющий вывод меню, состоящего из следующих пунктов :



* 1. текущий пользователь,
  2. объем используемой памяти,
  3. объем дискового пространства,
  4. запущенные процессы,
  5. процессы, запущенные текущим пользователем,
  6. системная дата и время,
  7. время запуска системы
  8. выход,

и вывод соответствующей информации в зависимости от выбранного пункта меню. Процесс повторять до тех пор, пока не будет выбран пункт меню «выход».

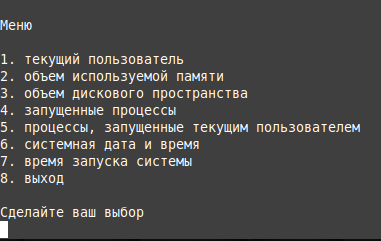




Добавим право на исполнение и запустим:



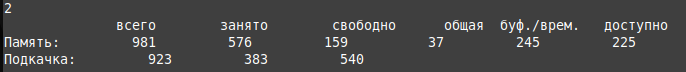
Для выбора в меню необходимо выбрать цифру, скрипт будет работать до тех пор, пока пользователь не выберет цифру 8.



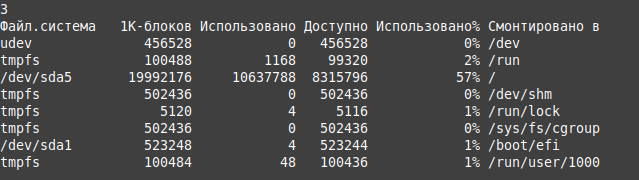
1. Определение текущего пользователя определяем через USER.



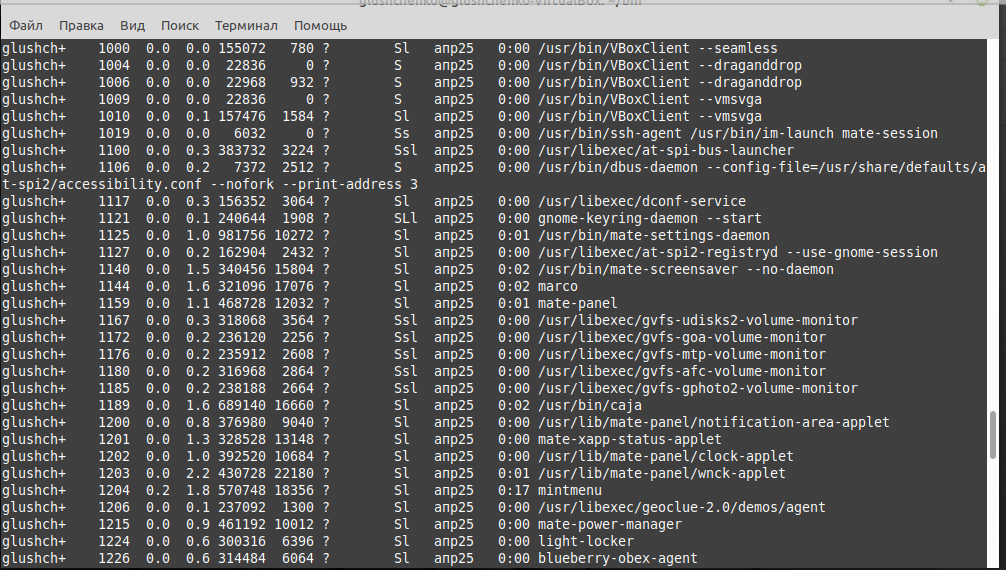
2. Чтобы найти объем используемой памяти, присвоим переменной MEM значение команды free -m, где -m это память в МБ.

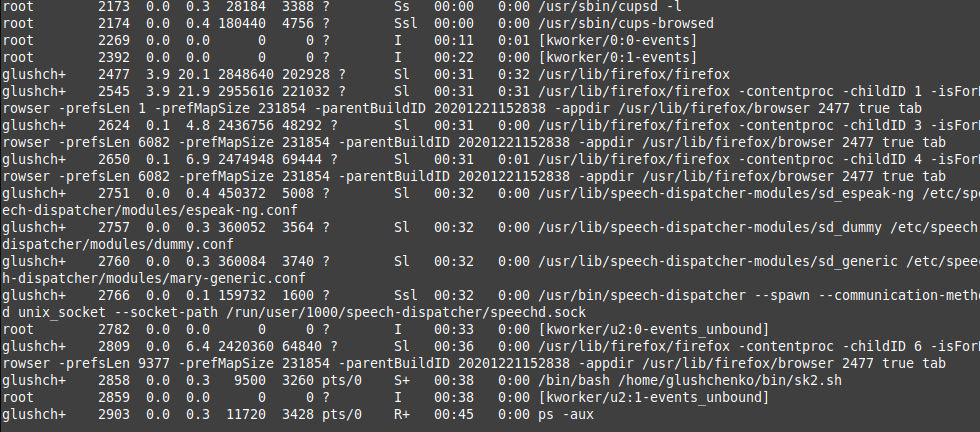


3. Чтобы найти объем дискового пространства, присвоим переменной DS значение команды df.

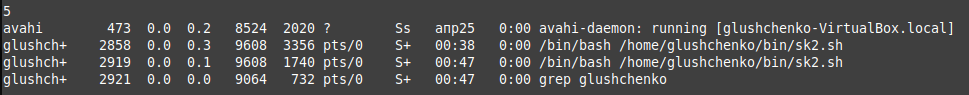


4. Чтобы определить запущенные процессы, присвоим переменной GS значение команды ps -aux.





5. Чтобы определить процессы, запущенные текущим пользователем процессы, присвоим переменной GPU значение команды ps -aux | grep $USER.



6. Чтобы определить cистемную дату и время, присвоим переменной DATEANDTIME значение команды date.



7. Чтобы определить время последнего запуска, присвоим переменной TSS значение команды who -b. Пример выполнения команды:



8. Чтобы выйти из скрипта, нужно выйти из цикла while, используем для этого break.



Полный текст сценария

echo "Меню"

echo

echo "1. текущий пользователь"

echo "2. объем используемой памяти"

echo "3. объем дискового пространства"

echo "4. запущенные процессы"

echo "5. процессы, запущенные текущим пользователем"

echo "6. системная дата и время"

echo "7. время запуска системы "

echo "8. выход"

echo

echo "Сделайте ваш выбор"

read yourchoise

while [ $per=="0" ]; do

if [ $yourchoise = 1 ]; then

echo "$USER"

echo

elif [ $yourchoise = 2 ]; then

MEM=$(free -m)

echo "$MEM"

echo

elif [ $yourchoise = 3 ]; then

DS=$(df)

echo "$DS"

echo

elif [ $yourchoise = 4 ]; then

GP=$(ps -aux)

echo "$GP"

echo

elif [ $yourchoise = 5 ]; then

GPU=$(ps -aux | grep $USER)

echo "$GPU"

echo

elif [ $yourchoise = 6 ]; then

DATEANDTIME=$(date)

echo "$DATEANDTIME"

echo

elif [ $yourchoise = 7 ]; then

TSS=$(who -b)

echo "$TSS"

echo

elif [ $yourchoise = 8 ]; then

break

else

echo "неверные данные"

fi;

read yourchoise

done