МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И. С. ТУРГЕНЕВА»

Кафедра информационных систем

**ОТЧЕТ**

по лабораторной работе № 3

на тему: «Cоздание и выполнение командных файлов в среде ОС Linux»

по дисциплине: «Операционные системы»

Выполнил: Кубарев А.Н.

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.03 «Прикладная информатика»

Группа: 71-ПИ

Проверил: Загородних Н.А.

Рыженков Д.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2017 г.

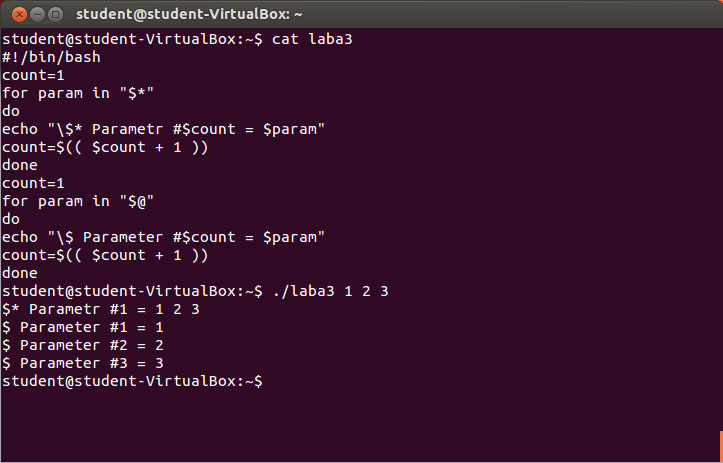
Орел, 2017 г.

**Цель работы**

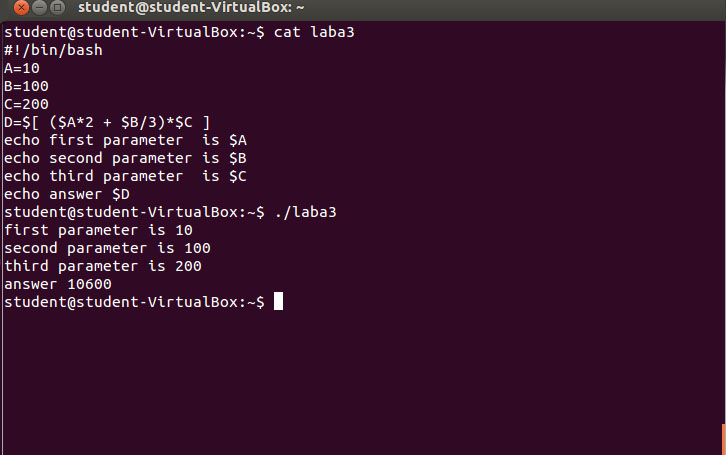
Целью работы является изучение методов создания и выполнения командных файлов на языке Shell - интерпретатора.

**Выполнение работы:**

1. Выведение на экран списка параметров командной строки с указанием номера каждого параметра.

****

1. Присвоение переменным А, В и С значений 10, 100 и 200, вычисление и вывод результатов по формуле D=(A\*2 + B/3)\*C.

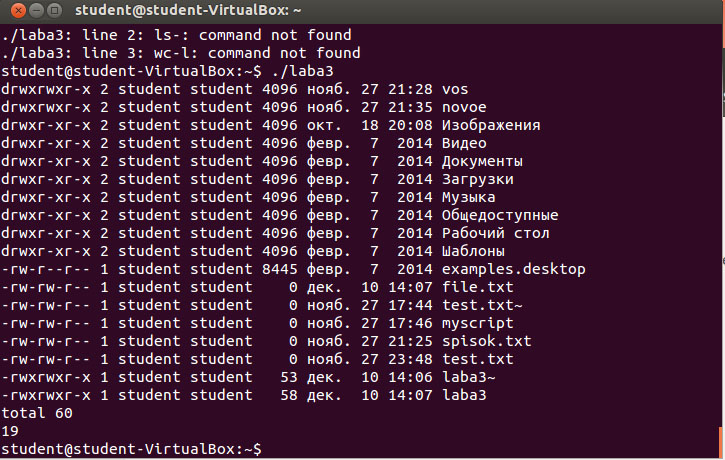


1. Формирование файла со списком файлов в домашнем каталоге, вывод на экран этого списка в алфавитном порядке и общего количества файлов.

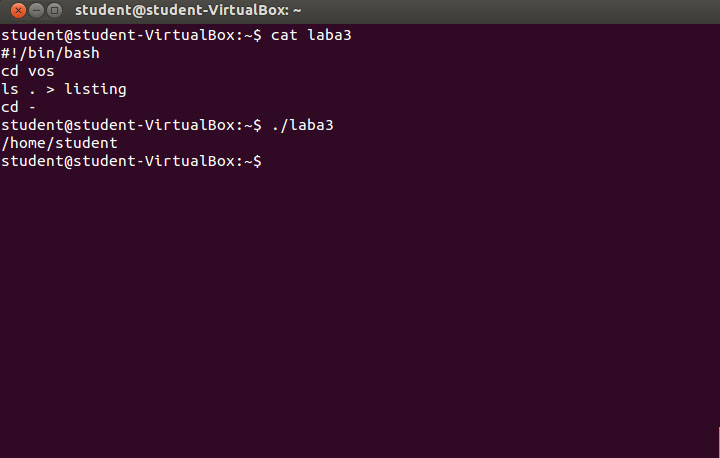
#!/bin/bash

ls -1 ~ |sort|tee spisok.txt

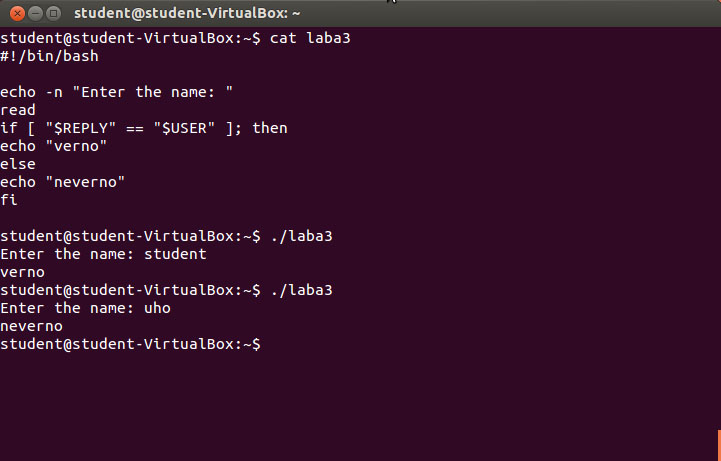
cat spisok.txt | wc -l



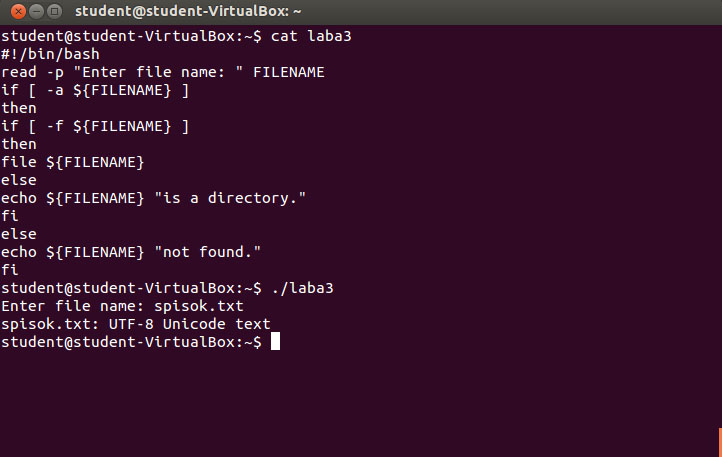
1. Переход в другой каталог, формирование файла с листингом каталога и возвращение в исходный каталог.



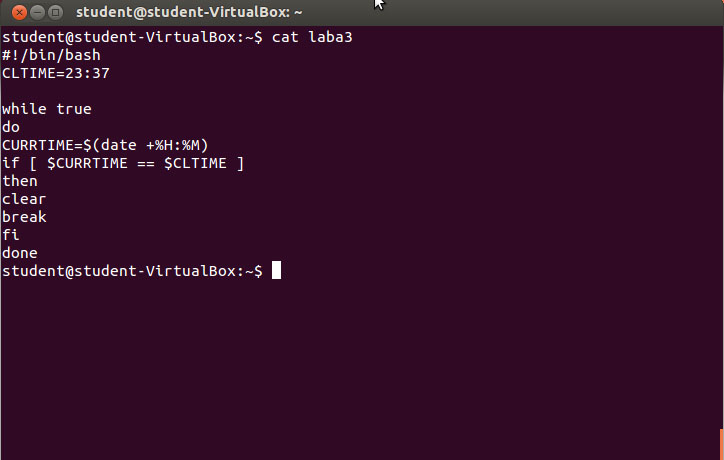
1. Запрос и ввод имени пользователя, сравнение с текущим логическим именем пользователя и вывод сообщения: верно/неверно.



1. Запрос и ввод имени файла в текущем каталоге и вывод сообщения о типе файла.



1. Циклическое чтение системного времени и очистка экрана в заданный момент.



8. Циклический просмотр списка файлов и выдача сообщения при появлении заданного имени в списке.

#!/bin/bash

ls -1

read -p "Enter file name:" FILENAME

for i in $(ls -1)

do

if [ ${i} == ${FILENAME} ]

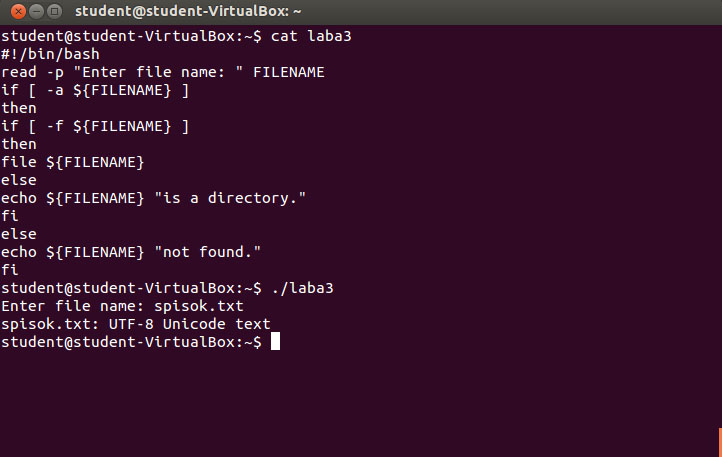
then

echo "expected file found:" $(pwd)"/""${i}"

exit 0

echo "file ${FILENAME} not found."

exit 1



Ответы на контрольные вопросы

1. Какое назначение имеют shell-файлы?   
   Командные оболочки, такие, как bash и Korn shell поддерживают программные конструкции, которые могут быть сохранены в форме сценариев. Эти сценарии, в свою очередь, впоследствии могут использоваться в качестве реализаций дополнительных команд командной оболочки. Многие команды Linux реализованы в форме сценариев. Например, сценарии для обслуживания профиля пользователя исполняются при входе пользователя в систему, а сценарии инициализации - при остановке и запуске демона.   
   Сценарии для командной строки — это файлы с простым текстом, которые интерпретируются командным процессором как наборы команд, подлежащие последовательному исполнению.
2. Как создать shell-файл и сделать его выполняемым?   
   Shell – файл Можно сделать выполняемым с помощью команды chmod u+x,r+x,o+x file. Также его можно выполнить, вызвав команду sh file или sh < file.   
   Файл можно выполнить и в текущем экземпляре Shell. Для этого существует специфическая команда "." (точка): . file.
3. Какие типы переменных используются в shell-файлах?   
   Переменные в Shell-файлах можно разделить на следующие группы:   
   1). Простые переменные, значения которых может задавать пользователь или они могут устанавливаться интерпретатором.   
   2). Позиционные переменные. Переменные вида $n, где n - целое число, используются для идентификации позиций элементов в командной строке с помощью номеров, начиная с нуля.   
   3). Специальные переменные. Переменные - ? # $ ! устанавливаются толькоShell. Они позволяют с помощью команды echo получить следующую информацию:   
   - – текущие флаги интерпретатора (установка флагов может быть изменена командой set);   
   # – число аргументов, которое было сохранено интерпретатором при выполнении какой-либо команды;   
   ? – код возврата последней выполняемой команды;   
   $ – числовой идентификатор текущего процесса PID;   
   ! – PID последнего фонового процесса.
4. В чем заключается анализ цепочки символов?   
   Shell содержит базовые средства для обработки цепочек символов. Так, вместо конструкции   
   ${#имя} подставляется число символов в значении переменной с заданным именем. Предоставляется четыре конструкции для сопоставления с образцом: ${имя%слово} ${имя%%слово} ${имя#слово} ${имя##слово} Во всех случаях слово рассматривается как образец (шаблон, см. далее); после сопоставления с ним подставляется значение переменной с заданным именем, из которого удален минимальный (максимальный) сопоставленный суффикс (префикс).
5. Какие встроенные команды используются в shell-файлах?   
   Встроенные команды являются частью интерпретатора и не требуют для своего выполнения проведения последовательного поиска файла команды и создания новых процессов. Встроенные команды:   
   cd [dir] - назначение текущего каталога;   
   exec [cmd [arg...]]<имя файла> - выполнение команды, заданной аргументами cmd и arg, путем вызова соответствующего выполняемого файла.   
   umask [ -o | -s] [nnn] - устанавливает маску создания файла (маску режимов доступа создаваемого файла, равную восьмеричному числу nnn: 3 восьмеричных цифры для пользователя, группы и других). Если аргумент nnn отсутствует, то команда сообщает текущее значение маски. При наличии флага -o маска выводится в восьмеричном виде, при наличии флага -s - в символьном представлении;   
   set, unset - режим работы интерпретатора, присваивание значений параметрам;   
   eval [ -arg] - вычисление и выполнение команды;   
   sh <filename.sh> - выполнение командного файла filename.sh;   
   exit [n] - приводит к прекращению выполнения программы, возвращает код возврата, равный нулю, в вызывающую программу;   
   trap [cmd] [cond] - перехват сигналов прерывания, где: cmd - выполняемая команда; cond=0 или EXIT - в этом случае команда cmd выполняется при завершении интерпретатора; cond=ERR - команда cmd выполняется при обнаружении ошибки; cond - символьное или числовое обозначение сигнала, в этом случае команда cmd выполняется при приходе этого сигнала;   
   export [name [=word]...] - включение в среду. Команда export объявляет, что переменные name будут включаться в среду всех вызываемых

впоследствии команд;   
wait [n] - ожидание завершения процесса. Команда без аргументов ожидает завершения процессов, запущенных синхронно. Если указан числовой аргумент n, то wait ожидает фоновый процесс с номером n;   
read name - команда вводит строку со стандартного ввода и присваивает прочитанные слова переменным, заданным аргументами name.

1. Как производится управление программами?   
   Команды true и false служат для установления требуемого кода завершения пpоцесса: true - успешное завершение, код завершения 0; false - неуспешное завершение, код может иметь несколько значений, с помощью которых определяется причина неуспешного завершения. Коды завершения команд используются для принятия решения о дальнейших действиях в операторах цикла while и until и в условном операторе if. Многие команды LINUX вырабатывают код завершения только для поддержки этих операторов.   
   Условный оператор if проверяет значение выражения. Если оно равно true, Shell выполняет следующий за if оператор, если false, то следующий оператор пропускается. Формат оператора if:   
   if <условие> then list1 else list2 fi   
   Команда test(проверить) используется с условным оператором if и операторами циклов. Действия при этом зависят от кода возврата test. Test проводит анализ файлов,   
   числовых значений, цепочек символов. Нулевой код выдается, если при проверке результат положителен, ненулевой код при отрицательном результате проверки.   
   В случае анализа файлов синтаксис команды следующий: test [ -rwfds] file, где   
   -r – файл существует и его можно прочитать (код завершения 0);   
   -w – файл существует и в него можно записывать;   
   -f – файл существует и не является каталогом;   
   -d – файл существует и является каталогом;   
   -s – размер файла отличен от нуля.   
   При анализе числовых значений команда test проверяет, истинно ли данное отношение, например, равны ли А и В . Сравнение выполняется в формате:   
   -eq А = В   
   -ne А <> B   
   test A -ge B эквивалентно А >= В   
   -le А <= В   
   -gt А > В   
   -lt А < В   
   Отношения слева используются для числовых данных, справа – для символов.   
   Кроме команды test, имеются еще некоторые средства для проверки:   
   ! - операция отрицания инвертирует значение выражения, например, выражение if test true эквивалентно выражению if test ! false;   
   o - двуместная операция "ИЛИ" (or) дает значение true, если один из операндов имеет значение true;   
   a - двуместная операция "И" (and) дает значение true, если оба операнда имеют значение true.
2. Назовите операторы создания циклов.   
   Оператор цикла с условием while true и while false. Команда while (пока) формирует циклы, которые выполняются до тех пор, пока команда while определяет значение следующего за ним выражения как true или false. Фоpмат оператора цикла с условием while true: while list1do list2 done. Здесь list1 и list2 - списки команд. While проверяет код возврата списка команд, стоящих после while, и если его значение равно 0, то выполняются команды, стоящие между do и done.   
   Оператор цикла с условием whilefalse имеет формат: until list1 do list2 done   
   В отличие от предыдущего случая условием выполнения команд между do и done является ненулевое значение возврата. Программный цикл может быть размещен внутри другого цикла (вложенный цикл). Оператор break прерывает ближайший к нему цикл. Если в программу ввести оператор break с уровнем 2 (break 2), то это обеспечит выход за пределы двух циклов и завершение программы.   
   Оператор continue передает управление ближайшему в цикле оператору while.   
   Оператор цикла с перечислением for: for name in [wordlist] do list done, где name - переменная; wordlist - последовательность слов; list - список команд. Переменная name получает значение первого слова последовательности wordlist, после этого выполняется список команд, стоящий между do и done. Затем name получает значение второго слова wordlist и снова выполняется список list. Выполнение прекращается после того, как кончится список wordlist.   
   Ветвление по многим направлениям case. Команда case обеспечивает ветвление по многим направлениям в зависимости от значений

аргументов команды. Формат:   
case <string> in   
s1) <list1>;;   
s2) <list2>;;   
sn) <listn>;;   
\*) <list>   
esac   
Здесь list1, list2 ... listn - список команд. Производится сравнение шаблона string с шаблонами s1, s2 ... sk ... sn. При совпадении выполняется список команд, стоящий между текущим шаблоном sk и соответствующими знаками.