Лабораторная работа №10: отчет.

Программирование в командном процессоре ОС UNIX. Командные файлы.

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.

Содержание

# Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

# Задание

1. Научится писать bash-скрипты.
2. Приобрести навыки по запуску и взаимодействию с bash-скриптами.

# Указание к работе

## Описание метода

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

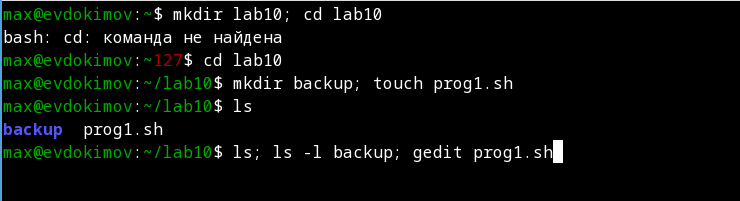
* оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
* С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
* оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
* BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

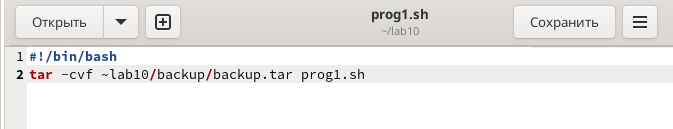
# Выполнение лабораторной работы

## Задание 1

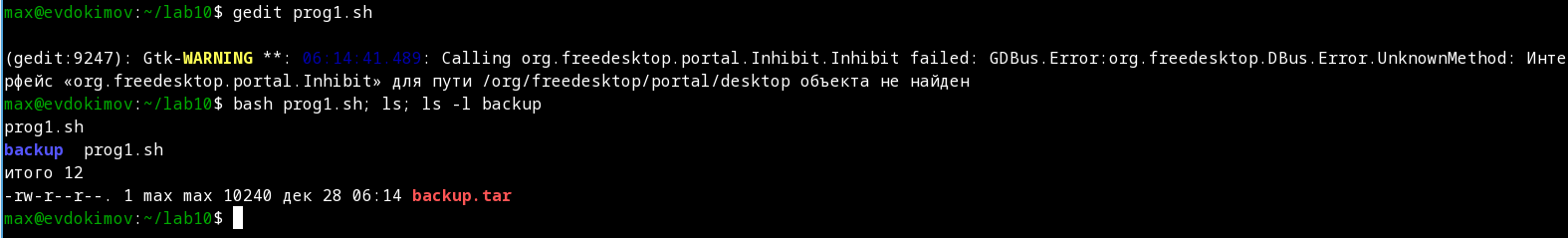
1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.



Создание файла prog1



Код файла prog1



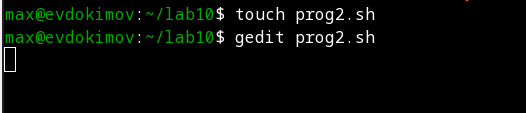
Результат запуска 1

## Код 1

#!/bin/bash  
tar -cvf ~/lab10/backup/backup.tar prog1.sh

## Задание 2

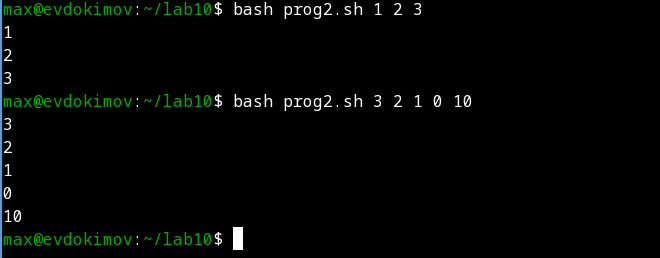
1. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.



Создание файла prog2



Код файла prog2



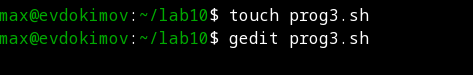
Результат запуска 2

## Код 2

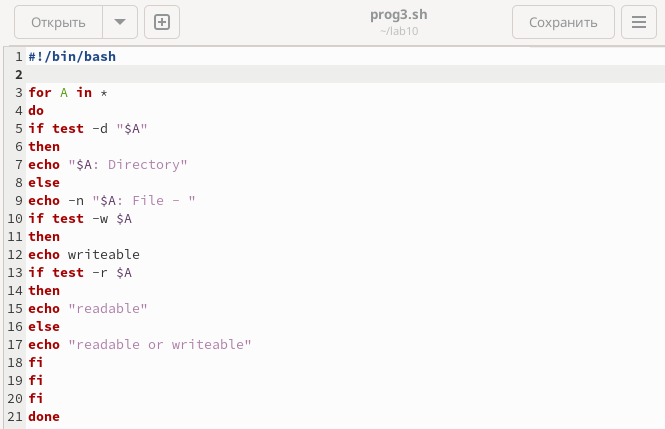
#!/bin/bash  
for A in $\*  
 do echo $A  
done

## Задание 3

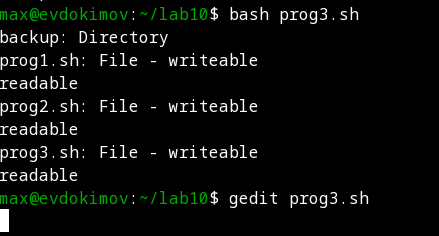
1. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.



Создание файла prog3



Код файла prog3



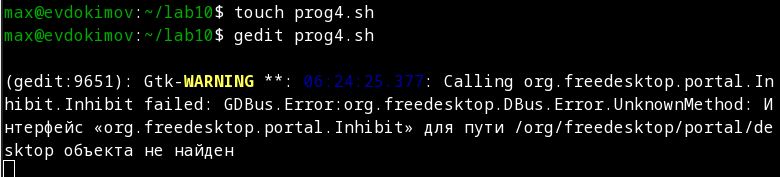
Результат запуска 3

## Код 3

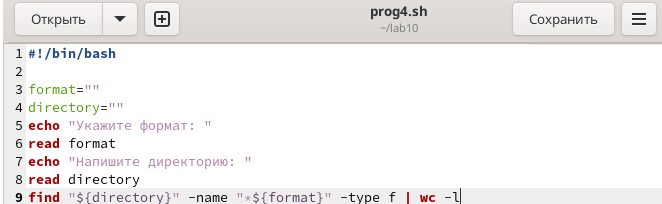
#!/bin/bash  
for A in \*  
do  
 if test -d "$A"  
 then  
 echo "$A: Directory"  
 else  
 echo -n "$A: File - "  
 if test -w $A  
 then  
 echo writeable  
 if test -r $A  
 then  
 echo "readable"  
 else  
 echo "readable or writeable"  
 fi  
 fi  
 fi  
done

## Задание 4

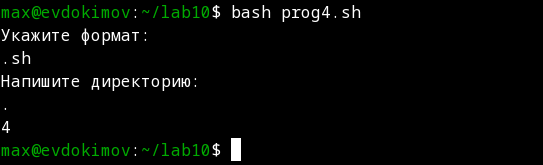
1. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.



Создание файла prog4



Код файла prog4



Результат запуска 4

## Код 4

#!/bin/bash  
format=""  
directory=""  
echo "Укажите формат: "  
read format  
echo "Напишите директорию: "  
read directory  
find "${directory}" -name "\*${format}" -type f | wc -l

# Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командные оболочки - это программы (процессы), позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам. UNIX обладает большим количеством оболочек. Наиболее популярными являются следующие четыре оболочки:

– оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

– С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

– оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;

– BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

1. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) - интерфейс переносимой операционной системы для компьютерных сред. Пред ставляет собой набор стандартов, подготовленных институтом инженеров по электронике и радиотехники (IEEE), который определяет различные аспекты построения операционной системы. POSIX включает такие темы, как программный интерфейс, безопасность, работа с сетями и графический интерфейс. POSIXсовместимые оболочки являются будущим поколением оболочек UNIX и других ОС. Windows NT рекламирует- ся как система, удовлетворяющая POSIXстандартам. POSIX-совместимые оболочки разработаны на базе оболочки Корна; фонд бесплатного про- граммного обеспечения (Free Software Foundation) работает над тем, чтобы и оболочку BASH сделать POSIX-совместимой.

1. Как определяются переменные и массивы в языке программирования bash?

Kомандный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда mark=/usr/andy/bin присваивает значение строки символов /usr/andy/bin переменной mark типа строка символов. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует мета символ.

1. Каково назначение операторов let и read?

Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение - это единичный терм (term), обычно целочисленный. Целые числа можно записывать как последовательность цифр или в любом базовом формате. Этот формат — radix#number, где radix (основание системы счисления) - любое число не более 26. Для большинства команд основания систем счисления это - 2 (двоичная), 8 (восьмеричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (\*), целочисленное деление (/) и целочисленный остаток (%). Команда let берет два операнда и присваивает их переменной.

1. Какие арифметические операции можно применять в языке программирования bash?

* “!” !ехр Если ехр равно 0, возвращает 1; иначе 0 != ехр1 !=ехр2 Если ехр1 не равно ехр2, возвращает 1; иначе 0;
* “%” ехр1%ехр2 возвращает остаток от деления ехр1 на ехр2 %= var=%exp присваивает остаток от деления var на ехр переменной var & ехр1&ехр2 возвращает побитовое;
* “AND” выражений ехр1 и ехр2 && ехр1 && ехр2 Если и ехр1 и ехр2 не равны нулю, возвращает 1; иначе 0 &= var &= ехр присваивает var побитовое AND переменных var;
* “*” выражения ехр* ехр1 \* ехр2 умножает ехр1 на ехр2 = var = ехр умножает ехр на значение var и присваивает результат переменной var + ехр1 + ехр2 Складывает ехр1 и ехр2 += var += ехр Складывает ехр со значением var и результат присваивает var;
* “-” -exp Операция отрицания exp (называется унарный минус) - expl - exp2 Вычитает exp2 из exp1 -= var -= exp вычитает exp из значения var и присваивает результат var;
* “/” exp / exp2 Делит exp1 на exp2 /= var /= exp Делит var на exp и присваивает результат var;
* “<” expl < exp2 Если exp1 меньше, чем exp2, возвращает 1, иначе возвращает 0 « exp1« exp2 Сдвигает exp1 влево на exp2 бит;
* “«=” var «= exp Побитовый сдвиг влево значения var на exp <= expl <= exp2 Если 16 exp1 меньше, или равно exp2, возвра- щает 1; иначе возвращает 0;
* “=” var = exp Присваивает значение exp переменной var;
* “==” exp1==exp2 Если exp1 равно exp2.Возвращает 1; иначе возвращает 0;
* “>” exp1 > exp2 1 если exp1 больше, чем exp2; иначе 0;
* “>=” exp1 >= exp2 1 если exp1 больше, или равно exp2; иначе 0 » exp » exp2 Сдвигает exp1 вправо на exp2 бит »= var »=exp Побитовый сдвиг вправо значения var на exp;
* “^” exp1 ^ exp2 Исключающее OR выражений exp1 и exp2 ^= var ^= exp присваивает var побитовое исключающее OR var и exp;
* “|” exp1 | exp2 Побитовое OR выражений exp1 и exp2 |= var |= exp Присваивает var «исключающее OR» переменой var и выражения exp;
* “||” exp1 || exp2 1 если или exp1 или exp2 являются ненулевыми значениями; иначе 0 ~ ~exp Побитовое дополнение до exp.

1. Что означает операция (( ))?

Условия оболочки bash.

1. Какие стандартные имена переменных Вам известны?

Имя переменной (идентификатор) — это строка символов, которая отличает эту переменную от других объектов программы (идентифицирует переменную в программе). При задании имен переменным нужно соблюдать следующие правила: § первым символом имени должна быть буква. Остальные символы — буквы и цифры (прописные и строчные буквы различаются). Можно использовать символ «\_»; § в имени нельзя использовать символ «.»; § число символов в имени не должно превышать 255; § имя переменной не должно совпадать с зарезервированными (служебными) словами языка. Var1, PATH, trash, mon, day, PS1, PS2 Другие стандартные переменные: –HOME — имя домашне- го каталога пользователя.

Если команда cd вводится без аргументов, то происходит переход в каталог, указанный в этой переменной . –IFS — последовательность символов, являющихся разделителями в командной строке. Это символы пробел, табуляция и перевод строки(new line). –MAIL — командный процессор каждый раз перед выводом на экран промптера про- веряет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение You have mail (у Вас есть почта). –TERM — тип используемого терминала. –LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему. В командном процессоре Си имеется еще несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды set.

1. Что такое метасимволы?

Такие символы, как “’” “<” “>” “\*” “?” “|” “”” “&” являются мета- символами и имеют для командного процессора специальный смысл.

1. Как экранировать метасимволы?

Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов, ее нужно заключить в одинарные кавычки. Строка, заклю- ченная в двойные кавычки, экранирует все метасимволы, кроме “$”, “’” , “\” , ““”. Например,–echo выведет на экран символ, –echo ab’|’cd выдаст строку ab|cd.

1. Как создавать и запускать командные файлы?

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде bash командный\_файл [аргументы] Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды chmod +x имя\_файла Теперь можно вызывать свой командный файл на выполнение просто, вводя его имя с терминала так, как будто он является выполняемой программой. Командный процессор распознает, что в Вашем файле на самом деле хранится не выполняемая программа, а про- грамма, написанная на языке программирования оболочки, и осуществит ее интерпретацию.

1. Как определяются функции в языке программирования bash?

Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключенных в фигурные скобки. Удалить функцию можно с помощью команды unset c флагом:

-f. Команда typeset имеет четыре опции для работы с функциями:

-f — перечисляет определенные на текущий момент функции;

-ft — при последующем вызове функции инициирует ее трассировку;

-fx — экспортирует все перечисленные функции в любые дочерние программы оболочек;

-fu — обозначает указанные функции как автоматически загружаемые.Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную FPATH, отыскивая файл с одноименными именами функций, загружает его и вызывает эти функции.

1. Каким образом можно выяснить, является файл каталогом или обычным файлом?

“ls -lrt” если есть d, то является файл каталогом.

1. Каково назначение команд set, typeset и unset?

Используется команда set с флагом -A. За флагом следует имя переменной, а затем список значений, разделенных пробелом. Например, set -A states Delaware Michigan “New Jersey”. Далее можно сделать добавление в массив, например, states=Alaska . Индексация массивов начинается с нулевого элемента. В командном про- цессоре Си имеется еще несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды set. Наиболее распространенным является сокращение, избавляющееся от слова let в программах оболочек. Если объявить переменные целыми значениями, любое присвоение автоматически трактуется как арифметическое. Используйте typeset -i для объявления и присвоения переменной, и при последующем использовании она становится целой. Или можете использовать ключевое слово integer (псевдоним для typeset -l) и объявлять переменные целыми. Таким образом, выражения типа х=y+z воспринимаются как арифметические. Группу команд можно объединить в функцию. Для этого существует ключевое слово function , после которого следует имя функции и список команд, заключенных в фигурные скобки. Удалить функцию можно с помо- щью команды unset c флагом -f . Команда typeset имеет четыре опции для работы с функциями: – -f — перечисляет определенные на текущий момент функции; – -ft — при последующем вызове функции инициирует ее трасси- ровку; – -fx — экспортирует все перечисленные функции в любые дочерние программы оболочек; – -fu — обозначает указанные функции как автома- тически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную FPATH , отыскивая файл с одноименными именами функций, загружает его и вызывает эти функции. В переменные mon и day будут считаны соответ- ствующие значения, введенные с клавиатуры, а переменная trash нужна для того, чтобы отобрать всю избыточно введенную информацию и игнори- ровать ее. Изъять переменную из программы можно с помощью команды unset.

1. Как передаются параметры в командные файлы?

Символ $ является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. Использование комбинации символов $0 приводит к подстановке вместо нее имени данного командного файла. Примере: пусть к команд- ному файлу where имеется доступ по выполнению и этот командный файл содержит следующий конвейер: “who | grep $1” Если Вы введете с терминала команду: where andy, то = в случае, если пользователь, зарегистрированный в ОС UNIX под именем andy,в данный момент работает в ОС UNIX, на терми- нал будет выведена строка, содержащая номер терминала, используемого указанным пользователем.

1. Назовите специальные переменные языка bash и их назначение.

$\* — отображается вся командная строка или параметры оболочки;

$? — код завершения последней выполненной команды;

$$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;

$! — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;

$- — значение флагов командного процессора;

${#} — возвращает целое число — количество слов, которые были результатом $;

${#name} — возвращает целое значение длины строки в переменной name;

${name[n]} — обращение к n-ному элементу массива;

${name[\*]} — перечисляет все элементы массива, разделенные пробелом;

${name[@]} — то же самое, но позволяет учитывать символы пробелы в самих переменных;

${name:-value} — если значение переменной name не определено, то оно будет заменено на указанное value;

${name:value} — проверяется факт существования переменной;

${name=value} — если name не определено, то ему присваивается значение value;

${name?value} — останавливает выполнение, если имя переменной не определено, и выводит value, как сообщение об ошибке; это выражение работает противоположно {name-value}. Если переменная определена, то подставляется value;

${name#pattern} — представляет значение переменной name с удаленным самым коротким левым образцом (pattern);

${#name[\*]} и ${#name[@]} — эти выражения возвращают количество элементов в массиве name $# вместо нее будет осуществлена подстановка числа параметров, указанных в командной строке при вызове данного командного файла на выполне.

# Выводы

В ходе выполнения лабораторной работы были изучены и повторены методы и принципы написания и запуска bash-скриптов.

# Список литературы

1. [Лабораторная работа №10](https://esystem.rudn.ru/mod/resource/view.php?id=970836)
2. [Основы bash скриптов](https://habr.com/ru/articles/726316/)
3. [Написание bash скриптов](https://losst.pro/napisanie-skriptov-na-bash)