Лабораторная работа
Написание сценариев.

1 Сценарии

Сценарий это текстовый файл содержащий команды интерпретатора. Как правило первая строка сценария имеет следующий вид: `#!/bin/sh` Данная строка означает, что при запуске сценария будет запускаться программа `/bin/sh` и ей в качестве стандартного ввода будет передан файл содержащий сценарий. Сценарий может быть запущен несколькими способами. В первом случае используется команда вида: `sh script.sh`, где sh командный интерпретатор, а script.sh имя файла сценария. Во втором случае непосредственно указывается имя сценария. При этом следует учитывать, что для запуска сценария он должен находиться в каталоге, указанном в переменной окружения PATH. В противном случае, кроме имени сценария необходимо указать, также, путь к нему. Также должны иметься права на запуск файла сценария.

2 Изменение прав доступа к файлу

Для изменения прав доступа к файлу предназначена команда `chmod`.
`chmod [-R]` *режим файл ...*
Меняет режим доступа к файлу. Режим может быть задан как в символьной форме, так и в цифровой. В символьной форме используются обозначения:

`u g o` — владелец, группа, прочие соответственно

`+ - =` — установить, убрать, оставить указанное право доступа

`r w x` — право на чтение, запись, выполнение файла

В цифровой форме используется восьмеричное число, получаемое сложением следующих значений для достижения нужного режима:

`4000` SUID присвоить процессу ID владельца файла при запуске.

`2000` SGID присвоить процессу ID группы при запуске.

`0400` RUSR присвоить право чтения файла владельцу

`0040` RGRP присвоить право чтения файла членам группы

`0004` ROTH присвоить право чтения файла прочим пользователям

`0200` WUSR присвоить право записи в файл владельцу

`0020` WGRP присвоить право записи в файл группе

`0002` WOTH присвоить право записи в файл прочим пользователям

`0100` XUSR присвоить право выполнения файла владельцу

`0010` XGRP присвоить право выполнения файла группе

`0001` XOTH присвоить право выполнения файла прочим пользователям

Параметр -R позволяет рекурсивно сменить режим доступа в подкаталогах. Примеры:
`chmod u+w,go-x file` добавить для владельца файла право на запись в файл, а для членов группы и прочих пользователей отменить право на выполнение файла.
`chmod 0755 file` установить для владельца все права на доступ к файлу, а для группы и прочих пользователей установить права только на чтение и выполнение файла.
`chmod 0640 file` установить для владельца файла права на запись в файл и на чтение из файла, для группы которой принадлежит файл право на чтение файла, прочим пользователям запретить доступ к файлу.

3 Параметры

Параметр — это объект хранящий значение. Параметр установлен если ему присвоено значение. Null является допустимым значением. Если параметр установлен, то он может перейти в состояние не установлен только при использовании встроенной команды unset. Параметр может быть обозначен именем, числом или специальным символом. *Переменная* — это параметр обозначенный именем.

3.1 Позиционные параметры

Позиционный параметр — это параметр обозначенный одной или более цифрами, отличными от единственной цифры 0. Если параметр обозначен несколькими цифрами, они должны заключаться в фигурные скобки. Позиционные параметры определяются аргументами указанными в командной строке при вызове оболочки. Позиционные параметры временно заменяются при вызове функций. Присвоение значений позиционным параметрам может быть произведено при помощи встроенной команды set. Если имеется сценарий revers.sh:

```
#!/bin/sh
echo $3
echo $2
echo $1
```

то при запуске с аргументами one, two, three:

```
reverse.sh one two three
```

сценарий выведет: three two one

4 Составные команды

Составные команды могут быть следующими:

`(список)`
: — список выполняется в подоболочке. Присвоения переменных и встроенные команды, оказывающие воздействие на окружение интерпретатора, не будут иметь силы после завершения выполнения команды. Код завершения равен коду завершения списка.

`{ список; }`
: — Список просто выполняется в текущей оболочке. Список должен прерываться символом завершения строки или точкой с запятой. Такая команда называется групповой. Код завершения равен коду завершения списка.

`(( выражение ))`
: Вычисляет значение арифметического выражения. Если значение выражения не равно нулю, то код завершения ноль, в противном случае код завершения равен единице.

  Арифметические выражения Оболочка позволяет вычислять арифметические выражения. Вычисления производятся с длинными целыми, без проверки переполнения, деление на ноль приводит к ошибке. Ниже перечислены арифметические операторы в порядке убывания приоритета.

`- +`	унарные плюс и минус
`! ~`	логическое и побитовое отрицание
`**`	возведение в степень
`* / %`	умножение, деление, остаток
`+ -`	сложение, вычитание

`< < > >`
: сдвиг битов

`<= >= < >`
: сравнение

`== !=` равенство, неравенство

`&` побитовое И

`^` побитовое исключающее ИЛИ

`|` побитовое ИЛИ

`&&` логическое И

`||` логическое ИЛИ

`expr1?expr2:expr3`
: условное выражение

`= *= /= %= += -= < <= > >= &= ^= |=`
: присвоение

В качестве операндов могут использоваться переменные. До вычисления выражения производится раскрытие параметров. Значение параметра приводится к длинному целому. Константы начинающиеся с 0 интерпретируются как восьмеричные числа, начинающиеся с 0x или 0X как шестнадцатеричные. Иначе числа записываются в форме `[base#]n`, где base — число от 2 до 64 определяющее систему счисления, а n число в этой системе. Если base опущено, то число считается десятичным. Цифры больше 9 представляются латинскими буквами в нижнем регистре, верхнем регистре, _, @, в данном порядке. Если основание не превышает 36, то большие и маленькие буквы равнозначны.

`[[выражение]]`
: Код завершения 0 или 1 зависит от значения условного выражения. Ноль если значение истина и единица в противном случае. Выражения могут объединяться следующими операторами, перечисленными в порядке убывания приоритета:

`( выражение )`
: возвращает значение выражения.

`! выражение`
: истина если выражение ложь.

`выражение1 && выражение2`
: истина если оба выражения истина.

`выражение1 || выражение2`
: истина если хотя бы одно из выражений истина.

Условные выражения Условные выражения используются в составной команде [[ ]] и во встроенных командах test и [ ]. Выражения формируются из следующих примитивов:

`-a file`
: Истина если файл существует

`-b file`
: Истина если файл существует и это файл блочно-ориентированного устройства

`-c file`
: Истина если файл существует и это файл байт-ориентированного устройства

`-d file`
: Истина если файл существует и это каталог

`-e file`
: Истина если файл существует

`-f file`
: Истина если файл существует и это регулярный файл

`-g file`
: Истина если файл существует и у него установлен бит SGID

`-h file`
: Истина если файл существует и это символическая ссылка

`-k file`
: Истина если файл существует и у него установлен бит SUID

`-p file`
: Истина если файл существует и это именованный канал

`-r file`
: Истина если файл существует и он доступен для чтения

`-s file`
: Истина если файл существует и его размер больше нуля

`-t fd` Истина если файл с указанным дескриптором открыт и это терминал

`-u file`
: Истина если файл существует и у него установлен бит SUID

`-w file`
: Истина если файл существует и доступен для записи

`-x file`
: Истина если файл существует и этот файл выполняемый

`-O file`
: Истина если файл существует и его владелец — пользователь чей идентификатор равен эффективному идентификатору выполняемого процесса

`-G file`
: Истина если файл существует и принадлежит группе идентификатор которой равен эффективному идентификатору группы выполняемого процесса

`-L file`
: Истина если файл существует и это символическая ссылка

`-S file`
: Истина если файл существует и это сокет

`-N file`
: Истина если файл существует и время изменения больше времени доступа

`file1 -nt file2`
: Истина если file1 новее file2

`file1 -ot file2`
: Истина если file1 старее file2

`-z string`
: Истина если длина строки ноль

`-n string`
: Истина если длина строки не ноль

`string1 == string2`
: Истина если строки равны

`string1 != string2`
: Истина если строки не равны

`string1 < string2`
: Истина если string1 при сортировке, с учетом текущей локали, окажется раньше string2

`string1 > string2`
: Истина если string1 при сортировке окажется после string2

`arg1 OP arg2`
: , где OP одна из следующих: -eq (равно), -ne (не равно), -lt (меньше чем), -le (меньше либо равно), -gt (больше чем), -ge (больше либо равно). Истина в случае выполнения соответствующего условия.

5 Операторы цикла

В оболочке имеются несколько операторов для организации циклов. Первый из них это оператор `for`:
`for` *имя* `[ in` *слово* `] ; do` *список* `; done`
Список слов, следующих за `in` раскрывается, образуя последовательность значений. Переменной *имя* поочередно присваиваются все эти значения, при этом каждый раз выполняется *список*. Если конструкция `in` *слово* опущена, список выполняется один раз для каждого установленного позиционного параметра. Код завершения равен коду завершения последней выполненой команды. Если раскрытие элементов следующих за `in` дает в результате пустой список, то выполнения команд не происходит и код завершения равен нулю.

`select` *имя* `[ in` *слово* `] ; do` *список* `; done`
Список слов, следующих за `in` раскрывается, образуя последовательность значений. Множество полученных значений выводится на стандартный вывод ошибок, каждое значение предваряется номером. Если конструкция `in` *слово* опущена, печатается список позиционных параметров. Затем выводится промпт `PS3` и считывается строка со стандартного ввода. Если строка содержит номер соответствующий одному из выведенных слов, то значение переменной *имя* устанавливается равным этому слову. Если строка пустая, то список печатается повторно. Если строка содержит `EOF` (вводится нажатием клавиш Ctrl-D), то цикл завершается. Любое другое значение приводит к тому, что переменная *имя* устанавливается в ноль. Полученная строка сохраняется в переменной `REPLY`. После каждого выбора выполняется список. Цикл повторяется до выполнения команды `break` или `return`. Код завершения равен коду завершения последней выполненой команды или нулю если команды не выполнялись.

`while` *список1;* `do` *список2;* `done`
`until` *список1;* `do` *список2;* `done`
Команда `while` выполняет *список2* пока *список1* завершается с кодом ноль. Команда `until` аналогична, но выполняет *список2* до тех пор, пока *список1* не завершится с кодом ноль. Код завершения команды равен коду завершения последней выполненой команды или нулю если ни одной команды не было выполнено.

6 Практическое задание

1. Создайте в домашнем каталоге подкаталог `bin`.
2. Выведите на экран значение переменной `PATH` и убедитесь, что она содержит созданный Вами каталог `bin`.
3. При помощи редактора vim создайте файл `bin/s1.sh`:

   ```
   for file ; do
   echo "-----$file-----"
   head -n 10 $file
   done
   ```

4. Измените права доступа к файлу. Для этого выполните команду

   ```
   chmod +x bin/s1.sh
   ```

   Команда добавляет разрешение на выполнение файла.
5. Выполните сценарий следующим образом:

   ```
   s1.sh /home/labs/text.txt /home/labs/dao.txt
   ```

6. Сохраните результат в файле отчёта.

7. Напишите сценарий, получающий в качестве первого аргумента имя каталога и выводящий список тех файлов из этого каталога, имена которых содержат последовательность символов определенную вторым аргументом. Сценарий споместитe в подкаталоге `bin` домашнего каталога и назовите `s2.sh`.

8. Напишите сценарий, который выводит пронумерованный список файлов с расширением `txt` в текущем каталоге, ожидает от пользователя ввода номера файла, выводит первые десять строк заданного файла и повторяет цикл. Сценарий сохраните под именем `list.sh` в каталоге `bin`.

9. Встроенная команда интерпретатора `read` *имя* считывает строку со стандартного ввода и записывает её в переменную *имя*. Команда возвращает код завершения ноль если строка была успешно считана и код отличный от нуля если произошла ошибка или встретился конец файла. Напишите сценарий, который считывает строки со стандартного ввода и выводит их на стандартный вывод. (Используйте цикл `while`). Сценарий сохраните под именем `s3.sh`.