

Лабораторная работа №10

Гэинэ Андрей

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

Задание

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.
2. 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.
3. 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.
4. Написать командный файл, который получает в качестве

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

```
zip scri1.zip scri1.txt
mv scri1.zip /home/andre/backup/

[andre@andre ~]$ chmod 777 scri1.txt
[andre@andre ~]$ ./scri1.txt
adding: scri1.txt (deflated 25%)
mv: невозможно переместить 'scri1.zip' в '/home/andre/backup/': Это не каталог
[andre@andre ~]$ mkdir backup
[andre@andre ~]$ ./scri1.txt
updating: scri1.txt (deflated 25%)
[andre@andre ~]$ ls
abcl      example  play      Видео      Общедоступные
australia feathers scri1.txt  Документы  'Рабочий стол'
backup    file.txt ski.places Загрузки   Шаблоны
conf.txt  fu       work      Изображения
etc       my_os    workisactuallyhere Музыка
```

2. 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.

```
scr2 [-M
for i
<----->do
<----->echo "$i"
done
```

```
7
[andre@andre ~]$ ./scr2 2 3 5 7 1 2 5 7 1 9 2 55
2
3
5
7
1
2
5
7
1
9
2
55
```

3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном к файлам каталога

```
ur=$(pwd)
if $(test -d $1); then cd $1
    for a in $(echo *)
        do echo "${a}"
            if $(test -w ${a}); then
                echo "read."
            fi

            if $(test -w ${a}); then
                echo "writ."
            fi

            if $(test -e ${a}); then
                echo "exec."
            fi
        done
    echo ""
else echo "arg is not directory"
fi
```

```
[andre@andre ~]$ ./scr3 backup
scr1.zip
read.
writ.
```

4. 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.

```
if (test -d $1); then
    echo $(ls $1 | grep "$2" | wc -l)
else echo "$1 isnt dir"
fi
```

```
[andre@andre ~]$ ./scr4 ~ .txt
3
```

Благодаря данной работе мы научились писать базовые программы.

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются? 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; - C-оболочка (или csh) — надстройка на оболочкой Борна, использующая подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка

2. Что такое POSIX? POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

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

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Потом значения переменных можно использовать в командах. Оболочка bash позволяет работать с массивами. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами.

4. Каково назначение операторов `let` и `read`? Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Команда `read` позволяет читать значения переменных со стандартного ввода.

5. Какие арифметические операции можно применять в языке программирования `bash`? Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/), целочисленный остаток от деления (%) и многие другие.

6. Что означает операция `(())`? Для облегчения программирования можно записывать условия оболочки `bash` в двойные скобки — `(())`.

7. Какие стандартные имена переменных Вам известны? Значением переменной PATH (т.е. \$PATH) является список каталогов, в которых командный процессор осуществляет поиск программы или команды, указанной в командной строке, в том случае, если указанное имя программы или команды не содержит ни одного символа /. Переменные PS1 и PS2 предназначены для отображения промптера командного процессора.

Другие стандартные переменные: - HOME — имя домашнего 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, например, пробел, табуляция и перевод строки (new line). - MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет 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

8. Что такое метасимволы? Такие символы, как ' < > * ? | " &, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола.

9. Как экранировать метасимволы? Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом.

10. 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

11. 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, заключённых в фигурные скобки.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом? `test имя_файла -d` выведет `true`, если файл каталог, и `false`, если он не каталог.

13. Каково назначение команд `set`, `typeset` и `unset`? Удалить функцию можно с помощью команды `unset` с флагом `-f`. Команда `typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определённые на текущий момент функции; `-ft` — при последующем вызове функции иницирует её 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 одноимёнными именами функций, загружает его и вызывает эти функции.

14. Как передаются параметры в командные файлы? При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. Символ `$` является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. При использовании где-либо в командном файле комбинации символов `$i`, вместо неё будет осуществлена подстановка значения параметра с порядковым номером `i`, т.е. аргумента командного файла с порядковым номером `i`

15. 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`; • `${name[n]}` — обращение к `n`-му элементу массива; • `${name[*]}` — перечисляет все элементы массива, разделённые 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;

- `${name+value}` — это выражение работает противоположно `${name-value}`. Если переменная определена, то подставляется value;
- `${name#pattern}` — представляет значение переменной name с удалённым самым коротким левым образцом (pattern);
- `${#name[*]}` и `${#name[@]}` — эти выражения возвращают количество элементов в массиве name.