

Лабораторная работа №10

Программирование в командном процессоре ОС UNIX. Командные
файлы

Гнатюк Анастасия Станиславовна

Содержание

Цель работы	3
Теоретическое введение	4
Выполнение лабораторной работы	5
Вывод	9
Контрольные вопросы	10

Цель работы

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

Теоретическое введение

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.

Выполнение лабораторной работы

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.

```
[asgnatyuk@fedora ~]$ emacs boom
```

Рис. 1: Рис.1:Запуск редактора emacs

```
File Edit Options Buffers Tools Help
tar -cf boom.tar boom
mv boom.tar ~/backup/boom.tar
```

Рис. 2: Рис.2:Код программы

```
[asgnatyuk@fedora ~]$ chmod +x boom
[asgnatyuk@fedora ~]$ ./boom
[asgnatyuk@fedora ~]$ cd backup
[asgnatyuk@fedora backup]$ ls
boom.tar
[asgnatyuk@fedora backup]$ tar -xf boom.tar
[asgnatyuk@fedora backup]$ ls
boom boom.tar
```

Рис. 3: Рис.3

2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Напри-

мер, скрипт может последовательно распечатывать значения всех переданных аргументов.

```
[asgnatyuk@fedora backup]$ emacs boom2
```

Рис. 4: Рис.4:Запуск редактора emacs

```
File Edit Options Buffers Tools Help
count=1
for param in "$@"
do
echo "$count: $param"
count=$((count + 1))
done
```

Рис. 5: Рис.5:Код программы

```
[asgnatyuk@fedora backup]$ emacs boom2
[asgnatyuk@fedora backup]$ ./boom2 1 2 3 4 5 6 7 8 9 10 11 12
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
10: 10
11: 11
12: 12
[asgnatyuk@fedora backup]$
```

Рис. 6: Рис.6

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.

```
[asgnatyuk@fedora backup]$ emacs boom3
```

Рис. 7: Рис.7:Запуск редактора emacs

```

File Edit Options Buffers Tools Help
for A in *
do if test -d $A
then echo $A:"Is a directory and "
else echo -n $A:"Is a file and "
if test -w $A
then echo writeable
elif test -r $A
then echo readable
else echo neither readable nor writeable
fi
fi
done

```

Рис. 8: Рис.8:Код программы

```

[asgnatyuk@fedora backup]$ emacs boom3
[asgnatyuk@fedora backup]$ chmod +x boom3
[asgnatyuk@fedora backup]$ ./boom3
boom:Is a file andwriteable
boom2:Is a file andwriteable
boom2~:Is a file andwriteable
boom3:Is a file andwriteable
boom.tar:Is a file andwriteable
[asgnatyuk@fedora backup]$ emacs boom3
[asgnatyuk@fedora backup]$ ./boom3
boom:Is a file and writeable
boom2:Is a file and writeable
boom2~:Is a file and writeable
boom3:Is a file and writeable
boom3~:Is a file and writeable
boom.tar:Is a file and writeable
[asgnatyuk@fedora backup]$

```

Рис. 9: Рис.9

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.

```

[asgnatyuk@fedora ~]$ emacs boom4

```

Рис. 10: Рис.10:Запуск редактора емас

```

File Edit Options Buffers Tools Help
echo "Input dir: "
read dirr
echo "Input file format: "
read form
find $dirr -name "$form" -type f | wc -l

```

Рис. 11: Рис.11:Код программы

```

[asgnatyuk@fedora ~]$ ./boom4
Input dir:
/home/asgnatyuk/
Input file format:
.txt
10
[asgnatyuk@fedora ~]$ ls
abc1 file.txt reports work
asgnatyuk.github.io gachi ski.places Видео
australia ghpages slide.txt Документы
backup hugo_extended_0.98.0_Windows-64bit test.txt Загрузки
boom logfile '#touch lab07.sh#' Изображения
boom- may 'touch lab07.sh' Музыка
boom4 monthly 'touch lab07.sh~' Общедоступные
boom4- my_os 'touch lab.sh#' 'Рабочий стол'
conf.txt newdir 'touch oops.sh#' Шаблоны
feathers placebo 'touch seeeeek.sh#'
file.c play vze
[asgnatyuk@fedora ~]$

```

Рис. 12: Рис.12

Вывод

Я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.

Контрольные вопросы

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;
- 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).

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?

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. Например,

```
1 set -A states Delaware Michigan "New Jersey"
```

Далее можно сделать добавление в массив, например, `states[49]=Alaska`. Индексация массивов начинается с нулевого элемента.

5. Какие арифметические операции можно применять в языке программирования `bash`?

`! expr` Если `expr` равно 0, то возвращает 1; иначе 0

`!= expr1 !=expr2` Если `expr1` не равно `expr2`, то возвращает 1; иначе 0

`% expr1%expr2` Возвращает остаток от деления `expr1` на `expr2`

`%= var=%expr` Присваивает остаток от деления `var` на `expr` переменной `var`

`& expr1&expr2` Возвращает побитовое AND выражений `expr1` и `expr2`

`&& expr1&&expr2` Если и `expr1` и `expr2` не равны нулю, то возвращает 1; иначе 0

`&= var &= expr` Присваивает переменной `var` побитовое AND `var` и `expr`

- `expr1 * expr2` Умножает `expr1` на `expr2`

`= var = expr` Умножает `expr` на значение переменной `var` и присваивает результат переменной `var`

`- expr1 + expr2` Складывает `expr1` и `expr2`

`+= var += expr` Складывает `expr` со значением переменной `var` и результат присваивает переменной `var`; и другие...

6. Что означает операция `(())`?

Подобно `C` оболочка `bash` может присваивать переменной любое значение, а произвольное выражение само имеет значение, которое может использоваться. При этом «ноль» воспринимается как «ложь», а любое другое значение выражения — как «истина». Для облегчения программирования можно записывать условия оболочки `bash` в двойные скобки — `(())`.

7. Какие стандартные имена переменных Вам известны?

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. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа `^`, который, в свою очередь, является метасимволом.

9. Как экранировать метасимволы?

Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме `$, ' , , "`.