Лабораторная работа №10

Программирование в командном процессоре ОС UNIX. Командные файлы

Марина Русланбековна Валиева

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 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 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.

3 Выполнение лабораторной работы

1. Написала скрипт, который при запуске делает резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в нашем домашнем каталоге. При этом файл архивируется одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации узнали, изучив справку.

```
mrvalieva@dk8n74 ~ $ man tar
mrvalieva@dk8n74 ~ $ touch script.sh
mrvalieva@dk8n74 ~ $ chmod +x script.sh
mrvalieva@dk8n74 ~ $ ./script.sh
mrvalieva@dk8n74 ~ $ cd work/study/2022-2023
mrvalieva@dk8n74 ~/work/study/2022-2023 $
```

```
1 #! /bin/bash
2 mkdir ~/backup
3 cp lab10_1.sh ~/backup/backup.sh
4 gzip ~/backup/backup.sh
```

```
mrvalieva@dk8n74 ~ $ touch lab10_1.sh
mrvalieva@dk8n74 ~ $ chmod +x lab10_1.sh
mrvalieva@dk8n74 ~ $ ls ~/backup/
```

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

Открыть *lab10_2.sh

```
1 #~/bin/bash
2 for i
3    do echo $1
4       shift
5 done
```

```
mrvalieva@dk8n74 ~ $ touch lab10_2.sh
mrvalieva@dk8n74 ~ $ chmod +x lab10_2.sh
mrvalieva@dk8n74 ~ $ ./lab10_2.sh M A R I 1 2
mrvalieva@dk8n74 ~ $ ./lab10_2.sh M A R I 1 2
M
A
R
I
1
2
```

3. Написали 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.

```
mrvalieva@dk8n74 ~ $ touch lab10_3.sh
mrvalieva@dk8n74 ~ $ chmod +x lab10_3.sh
mrvalieva@dk8n74 ~ $ ./lab10_3.sh ~/work/
READ
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/study
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/blog
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/mrvalieva.g
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/os
WRITE
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/study
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrvalieva/work/blog
```

Рис. 3.1: задание 3

4. Написали 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.

```
#!/bin/bash
direct=''
form=''
echo 'write format'
read form
echo 'write directory'
read direct
find "$direct" -name "*.$form" -type f | wc -l
ls
```

```
mrvalieva@dk8n74 ~ $ touch lab10_4.sh
mrvalieva@dk8n74 ~ $ chmod +x lab10_4.sh
mrvalieva@dk8n74 ~ $ ./lab10_4.sh
write format
sh
write directory
backup
0
 1.py      GNUstep      lab10_2.sh   public_html   Видео          Общедоступные
 2.py      hello.asm    lab10_3.sh   __pycache__   Документы     'Рабочий стол'
 backup    lab09.sh     lab10_4.sh   script.sh     Загрузки       Шаблоны
 bin       lab09.sh~    my_os        tmp           Изображения
 Desktop   lab10_1.sh   public       work          Музыка
```

4 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?
2. Что такое POSIX?
3. Как определяются переменные и массивы в языке программирования bash?
4. Каково назначение операторов let и read?
5. Какие арифметические операции можно применять в языке программирования bash?
6. Что означает операция (( ))?
7. Какие стандартные имена переменных Вам известны?
8. Что такое метасимволы?
9. Как экранировать метасимволы?
10. Как создавать и запускать командные файлы?
11. Как определяются функции в языке программирования bash?
12. Каким образом можно выяснить, является файл каталогом или обычным файлом?
13. Каково назначение команд set, typeset и unset?
14. Как передаются параметры в командные файлы?
15. Назовите специальные переменные языка bash и их назначение.

5 Ответы на контрольные вопросы

1.

a) sh — стандартная командная оболочка UNIX/Linux, содержащая базовый, полный набор функций
b) csh — использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд
c) ksh — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна
d) bash — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна

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

3. Переменные вызываются $var, где var=чему-то, указанному пользователем, неважно что бы то не было, название файла, каталога или еще чего. Для массивов используется команда set -A

4. let — вычисляет далее заданное математическое значение read — позволяет читать значения переменных со стандартного ввода

5. Прибавление, умножение, вычисление, деление), сравнение значений, экспонирование и др.

6. Это обозначение используется для облегчения программирования для условий bash
7. Нам известны HOME, PATH, BASH, ENV, PWD, UID, OLDPWD, PPID, GROUPS, OSTYPE, PS1 - PS4, LANG, HOSTFILE, MAIL, TERM, LOGNAME, USERNAME, IFS и др.
8. Метасимволы это специальные знаки, которые могут использоваться для сокращения пути, поиска объекта по расширению, перед переменными, например «$» или «*» .
9. Добавить перед метасимволом метасимвол «»
10. При помощи команды chmod. Надо дать права на запуск chmod +x название файла, затем запустить bash ./название файла Например у нас файл lab Пишем: chmod +x lab ./lab
11. Объединяя несколько команд с помощью function
12. Можно задать команду на проверку диретория ли это test -d директория
13. Set — используется для создания массивов Unset — используется для изъятия переменной Typeset — используется для присваивания каких-либо функций
14. Добавлением аршументов после команды запуска bash скрипта
15.

- $* – отображается вся командная строка или параметры оболочки;
- $? – код завершения последней выполненной команды;
- $$ – уникальный идентификатор процесса, в рамках которого выполняется командн
- $! – номер процесса, в рамках которого выполняется последняя вызванная на вып
- $- – 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 пере
- ${name:-value}–еслизначениепеременнойnameнеопределено,тоонобудет заменено на
- ${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,
- ${name+value} – это выражение работает противоположно ${name-
value}. Если переменная определена, то подставляется value;
- ${name#pattern} – представляет значение переменной name с удалённым самым кор
- ${#name[*]}и${#name[@]}–этивыражениявозвращаютколичествоэлементов в массиве n

6 Выводы

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