Российский универститет дружбы народов

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе № 11

дисциплина: Операционные системы

Студент: Гаглоев Олег Мелорович

Преподаватель: Велиева Т.В.

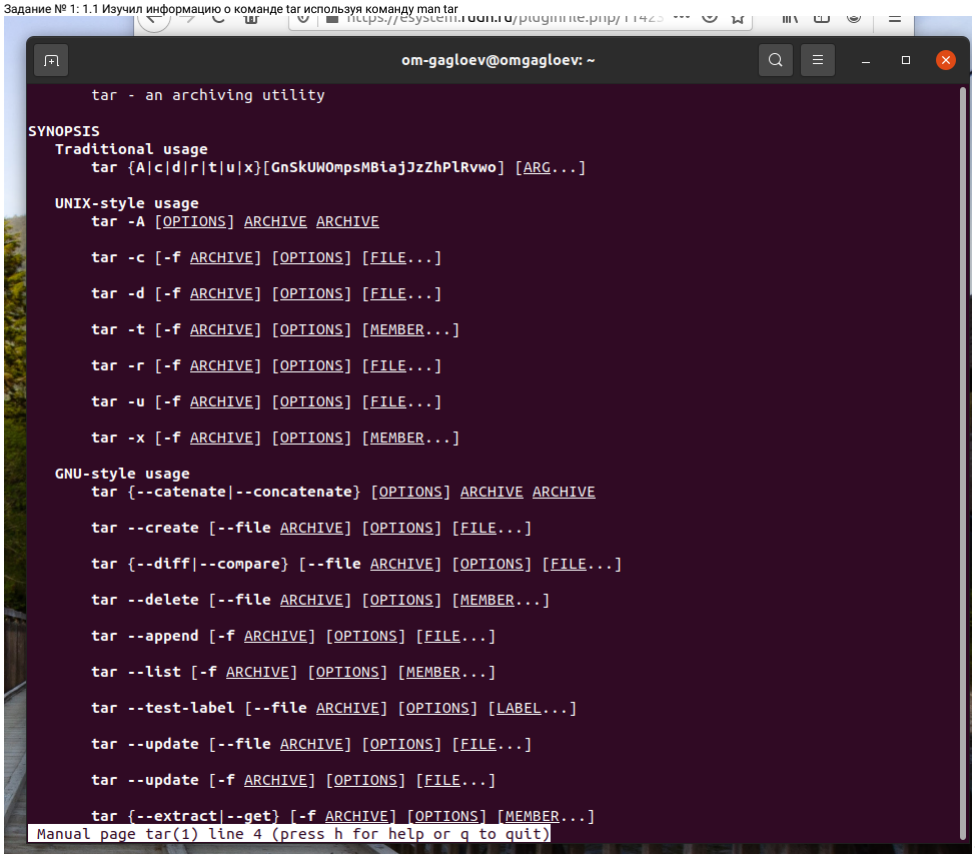
Группа НПИбд-01-20

```
                Москва 2021 г.
```

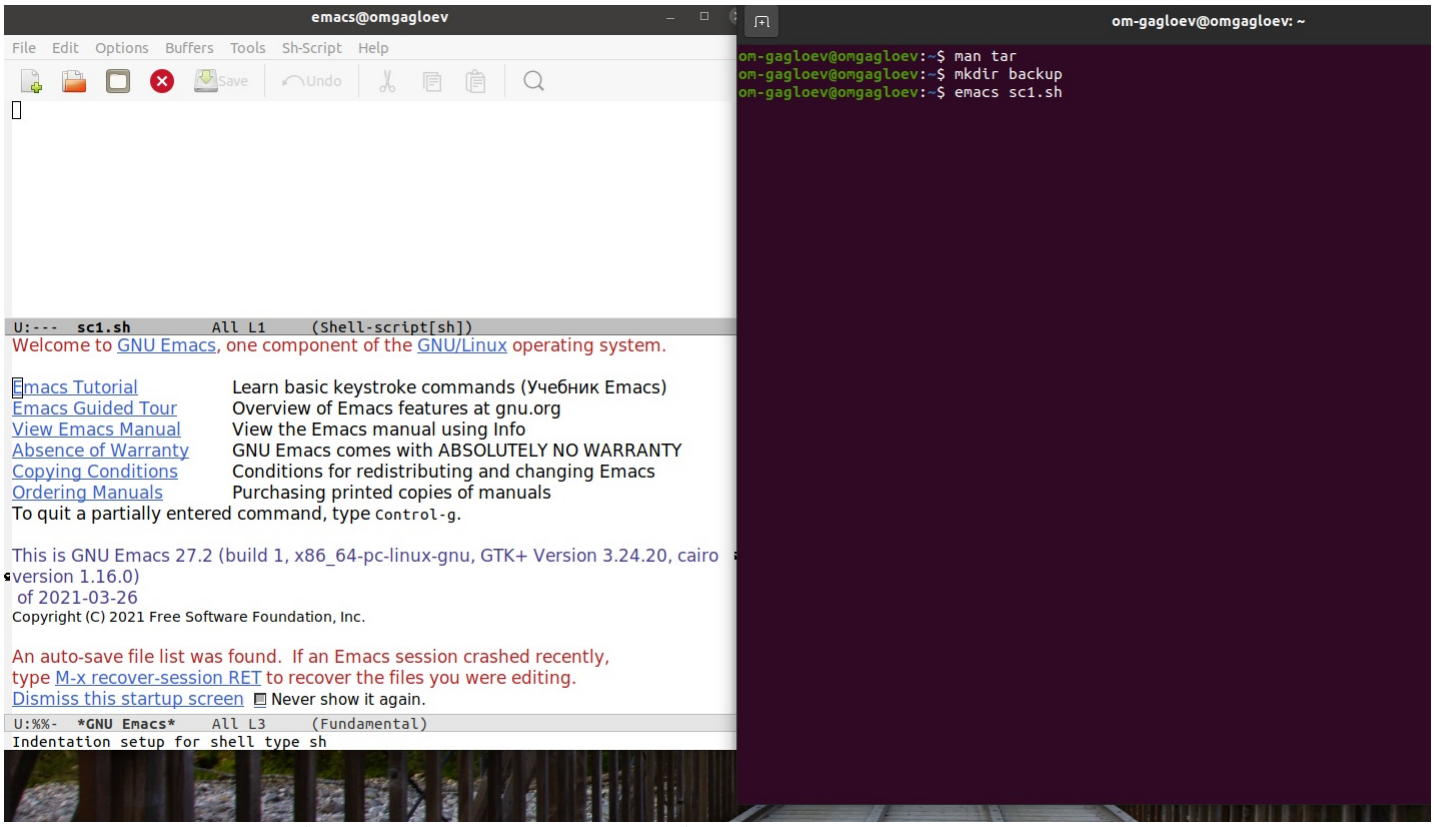
Цель работы : Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

Выполнение работы :

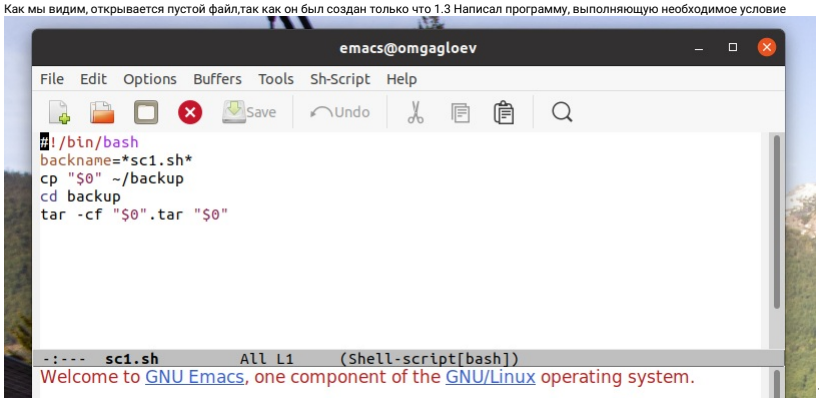
Задание № 1: 1.1 Изучил информацию о команде tar используя команду man tar



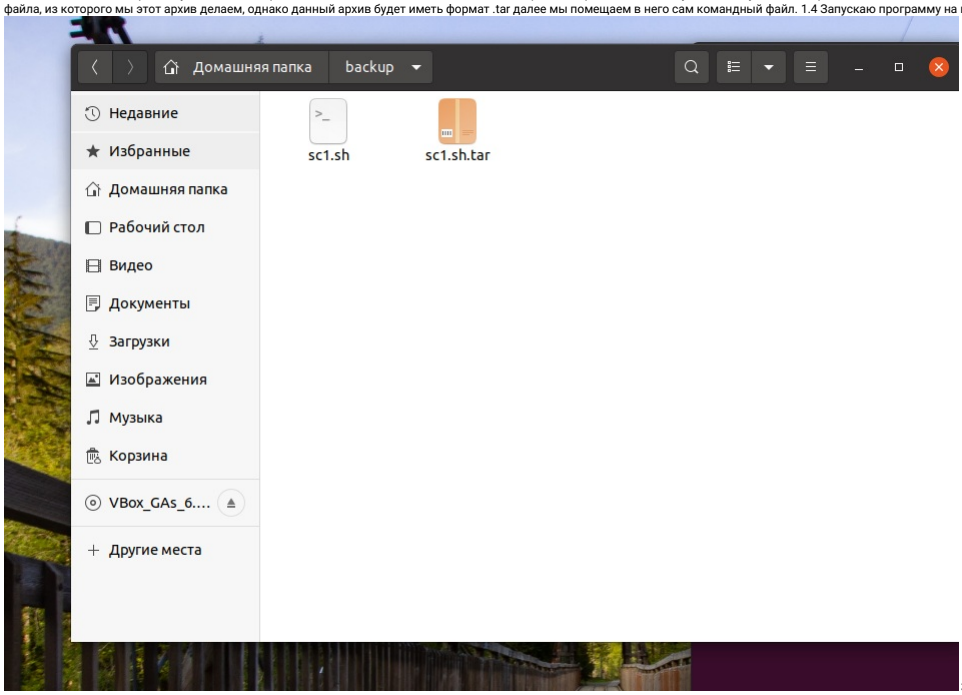
1.2 Создал директорию backup в моём домашнем каталоге , используя команду mkdir. Теперь создадим файл sc1.sh в текстовом редакторе emacs с помощью команды emacs sc1.sh



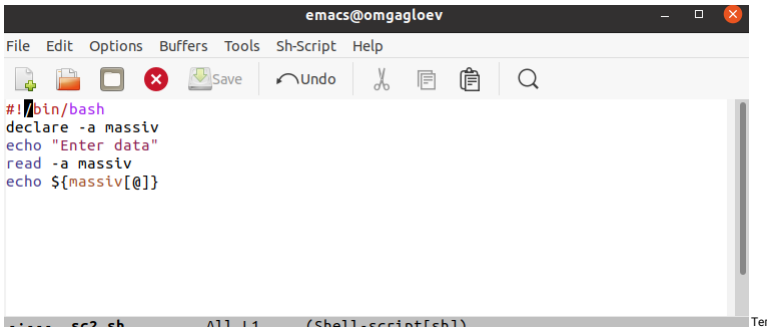
Как мы видим, открывается пустой файл,так как он был создан только что 1.3 Написал программу, выполняющую необходимое условие



Теперь объясню , что написанно в файле. Самая первая строка вызывает интерпритатор bash. После этого мы объявляем переменную backname , при этом помещаем в неё название sc1.sh На третьей строке мы используем команду cd, чтоб перейти в сам каталог и на последней строке использую команду tar -cf,чтобы создать архив с именем файла, из которого мы этот архив делаем, однако данный архив будет иметь формат .tar далее мы помещаем в него сам командный файл. 1.4 Запускаю программу на выполнение, введя в командной строке bash sc1.sh и проверяю результат



Задание № 2: 2.1 Написал программу, выполняющую необходимые действия



Теперь объясню, что написано в файле Самая первая строка вызывает интерпритатор bash. Далее мы объявляем массив massiv и командой echo вывожу текст После этого считываю данные, введённые пользователем с клавиатуры. и в последней строке вывожу все элементы массива 2.2 Запустим файл с помощью команды bash sc2.sh и введем несколько наборов чисел (

```
om-gagloev@omgagloev:~$ emacs sc2.sh
om-gagloev@omgagloev:~$ bash sc2.sh
Enter data
1 2 5 6 81 20 25
1 2 5 6 81 20 25
```

```
Enter data
1 2 3 4 5 6 91 20 36 58 78 74 51 203
1 2 3 4 5 6 91 20 36 58 78 74 51 203
```

Видим, что программа вывела наши элементы, даже когда их количество было больше 10 Задание № 3: 3.1)Написал программу, которая выполняет необходимые действия

```
#!/bin/bash
echo "Введите полное имя каталога"
read nk
cd ${nk}
echo "Содержимое данного каталога и права доступа :"
for A in *
do
echo $A
if test -r $A
then echo ' доступен для чтения'
if test -w $A
then echo 'доступен для записи '
else echo 'не доступен для записи и чтения'
fi
fi
done
```

Теперь объясню, как она работает: в первой строке мы вызываем интерпритатор bash.С помощью команды echo прошу ввести имя каталога. Затем считываю имя каталога с клавиатуры в переменную nk. Далее переходим в него командой cd.Далее идет цикл, который выполняется для каждого файла (A) в текущем каталоге (*). В цикле мы выводим название файла командой echo $A. Затем условием if test -r A проверяется есть ли право на чтение этого файла и после этого команда эхо echo выводит информацию о состоянии файла.Аналогично проверяется право на запись. Запускаю программу с помощью команды bash sc3.sh.Видим, что у нас появился запрос на ввод директории В качестве директории возьму ту, что мы создали в первом задании

```
om-gagloev@omgagloev:~$ emacs sc3.sh
om-gagloev@omgagloev:~$ bash sc3.sh
Введите полное имя каталога
backup
Содержимое данного каталога и права доступа :
sc1.sh
 доступен для чтения
доступен для записи
sc1.sh.tar
 доступен для чтения
доступен для записи
om-gagloev@omgagloev:~$
```

Видим, что у программа вывела нам названия файлов данного каталога и права доступа Задание № 4: Пишу программу, выполняющую задание

```
#!/bin/bash
dir=""
form=""
echo "Enter directory"
read dir
echo "Enter files format"
read form
cd ${dir}
find -name "*.$form" | wc -l
```

Теперь объясню её структуру: в первой строке мы вызываем интерпритатор bash. Объявляем две переменные: для хранения имени директории dir и искомого формата form. Далее мы выводим фразы запроса и считываем имя директории и искомый формат с клавиатуры с помощью read. Далее мы переходим в нужную директорию. И в последней строке мы совершаем поиск: файлы по именам find -name, в которых встречается нам введенный формат. Конвейером считываем нереализованный вывод и командой wc -l считаем его строки, т.е. - файлы, найденные в данной директории и соответствующие требованиям Запускаю программу с помощью команды bash sc4.sh

```
om-gagloev@omgagloev:~$ emacs sc4.sh
om-gagloev@omgagloev:~$ bash sc4.sh
Enter directory
backup
Enter files format
sh
1
om-gagloev@omgagloev:~$ bash sc4.sh
Enter directory
backup
Enter files format
txt
0
```

Видим, что программа действительно работает

Вывод:Я изучил основы программирования в оболочке ОС UNIX/Linux. Научился писать небольшие командные файлы.

Контрольные вопросы :

Командные процессоры или оболочки - это программы, 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. Примеры: Оболочка Борна (Bourne) - первоначальная командная оболочка UNIX: базовый, но полный набор функций; C - оболочка - добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя C-подобный синтаксис команд, и сохраняет историю выполненных команд; Оболочка Корна - напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; BASH - сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments)- интерфейс переносимой операционной системы для компьютерных сред. Представляет 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. POSIX-совместимые оболочки являются будущим поколением оболочек UNIX и других ОС. Windows NT рекламируется как система, удовлетворяющая POSIX-стандартам. POSIX-совместимые оболочки разработаны на базе оболочки Корна; фонд бесплатного программного обеспечения (Free Software Foundation) работает над тем, чтобы и оболочку BASH сделать POSIX-совместимой. Командный 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 метасимвол .Например,команда *mvafilemark* переместит файл afile из текущего каталога в каталог с абсолютным полным именем /usr/andy/bin. Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того, чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи: имяпеременнойнапример,использованиекоманд*b=/tmp/andy-ls-lmyfile>{b}ls* приведет к переназначению стандартного вывода команды ls с терминала на файл /tmp/andy-ls , а использование команды ls -l>$bls приведет к подстановке в командную строку значения переменной bls. Если переменной bls не было предварительно присвоено никакого значения, то ее значением является символ пробел. Оболочка bash позволяет создание массивов. Для создания массива 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[49]=Alaska. Индексация массивов начинается с нулевого элемента. 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 (восьме- ричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), *целочисленное деление (/) и целочисленный остаток (%). Команда let берет два операнда и присваивает их переменной.*

*Команда read читает одну строку из стандартного входного потока и записывает ее содержимое в указанные переменные. Если задана единственная переменная, в нее записывается вся строка. В результате ввода команды read без параметров строка помещается в переменную среды $reply. При указании нескольких переменных в первую из них записывается первое слово строки, во вторую — второе слово и т. д. Последней переменной присваивается остаток строки. "+" сложение "-" вычитание " умножение "/" деление "**" возведение в степень "%" остаток от деления Внутри (( )) вычисляются арифметические выражения и возвращается их результат. Например, в простейшем случае, конструкция a=$((5 + 3)) присвоит переменной «a» значение выражения «5 + 3», или 8. Кроме того, двойные круглые скобки позволяют работать с переменными в стиле языка C. В переменной $BASH содержится полный путь до исполняемого файла командной оболочки Bash. В переменную $BASH_VERSION записывается версия Bash. Переменная, которая хранит пути поиска каталога. (используется при вводе команды cd имя*каталога без слэша) *Содержит список каталогов для поиска файлов классов Java и архивов Java. Домашний каталог текущего пользователя. В переменной $HOSTNAME хранится имя компьютера. Количество событий, хранимых в истории за 1 сеанс Расположение файла истории событий Количество событий, хранимых в истории между сеансами Переменная хранит символы, являющиеся разделителями команд и параметров. (по умолчанию - пробел, табуляция и новая строка) Текущая установка локализации, которая позволяет настроить командную оболочку для использования в различных странах и на различных языках. Место, где храниться почта В переменной $OSTYPE содержится описание операционной системы. Список каталогов для поиска команд и приложений, когда полный путь к файлу не задан. PS1 используется как основная строка приглашения. (то самое [root@proxy ~]#) PS2 используется как вторичная строка приглашения. Эта команда должна быть выполнена до отображения строки приглашения Bash. Полный путь к текущему рабочему каталогу. Полный путь к текущей командной оболочке. В переменной $USER содержится имя текущего пользователя. Такие символы, как ' < > * ? | " & . Они имеют для командного процессора специальный смысл. 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, заключенная в двойные кавычки, экранирует все метасимволы, кроме $, ' , ". Например,–echo *выведет на экран символ,–echo ab'|'cdвыдаст строку ab|*cd. Для создания командного файла: Запустить текстовый редактор. Последовательно записать команды, располагая каждую команду на отдельной строке. Сохранить этот файл, сделать его исполняемым, применив команду: chmod +x имя*файла. Запустить этот файл можно или используя команду bash имя файла 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 с 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. ls -lrt Если есть d, то является файл каталогом Команда set изменяет значения внутренних переменных сценария. Команда typeset задаёт и/или накладывает ограничения на переменные. Команда unset удаляет переменную, фактически – устанавливает ее значение в null. В командный файл можно передать до девяти параметров. При использовании где-либо в команд- ном файле комбинации символов $i, где 0 < i < 10, вместо нее будет осуществлена подстановка значения параметра с порядковым номером i, т.е. аргумента командного файла с порядковым номером i. Использование комбинации символов $0 приводит к подстановке вместо нее имени данного командного файла. Рассмотрим это на примере. Пусть к командному 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 терминал будет выведена строка, содержащая номер терминала, используемого указанным пользователем. Если же в данный момент этот пользователь не работает в ОС UNIX, то на терминал не будет выведено ничего. Команда grep производит контекстный поиск в тексте, поступающем со стандартного ввода, для нахождения в этом тексте строк, содержащих последовательности символов, переданные ей в качестве аргументов, и выводит результаты своей работы на стандартный вывод. В этом примере команда grep используется как фильтр, обеспечивающий ввод со стандартного ввода и вывод всех строк, содержащих последовательность символов andy, на стандартный вывод. В ходе интерпретации этого файла командным процессором вместо комбинации символов 1осуществляетсяподстановказначенияпервогоиединственногопараметраandy.Еслипредположить,чтопользователь,зарегистрированныйвОСUNIXподименемandy,вданныймоментработаетвОСUNIX,тонатерминалеВыувидитепримерноследующее: where andy andy ttyG Jan 14 09:12 $ Определим функцию, которая изменяет каталог и печатает список файлов: $ function clist { > cd $1 > ls > }. Теперь при вызове команды clist каталог будет изменен каталог и выведено его содержимое. $* — 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 про- белом; ${name[@]} — то же самое, но позволяет учитывать символы про- белы 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—этовыражениеработаетпротивоположно{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 выполнение.