МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра “Информационные технологии”

Отчет по лабораторной работе № 4

«Shell-программирование»

по дисциплине «Операционные системы»

Выполнила: студент гр. ИТП-11

Валуев В.В.

Принял: преподаватель

Карась В.О.

Гомель 2022

***Цель работы*:** изучить синтаксис и операторы для языка Shell, научиться экспортировать локальные переменные в среду Shell и управлять ими, получить навыки в построении циклов и ветвлений вычислительных процессов, изучить проверку условий.

***Теоретические сведения:***

**ОПЕРАТОРЫ – КОМАНДЫ**

Язык shell по своим возможностям приближается к высокоуровневым алгоритмическим

языкам программирования. Операторы языка shell позволяют создавать собственные

программы, не требует компиляции, построения объектного файла и последующей

компоновки, так как shell, обрабатывающий их, является транслятором

интерпретирующего, а не компилирующего типа.

Текст процедуры набирается как обычный текстовый файл. Проверенный и отлаженный

shell-файл может быть вызван на исполнение, например, следующим способом:

$ chmod u+x shfil

$ shfil

$

Такая форма предполагает, что файл процедуры новый и его надо сначала сделать

выполняемым. Можно использовать также и следующий способ:

$ sh -c "shfil" или $ sh shfil

В этих случаях по команде sh вызывается вторичный интерпретатор shell, и в качестве

аргумента ему передается командная строка, содержащая имя файла процедуры shfil,

находящегося в текущем каталоге. Однако, этот способ накладывает ограничения на

исполнение некоторых команд ОС управления процессами.

Процедуре при ее запуске могут быть переданы аргументы. В общем случае командная

строка вызова процедуры имеет следующий вид:

$ имя\_процедуры $1 $2 ...$9

Каждому из девяти первых аргументов командной строки в тексте процедуры

соответствует один из позиционных параметров: $1, $2, ..., $9 соответственно. Параметр $0

соответствует имени самой процедуры, т.е. первому полю командной строки. К каждому из

10 первых аргументов можно обратиться из процедуры, указав номер его позиции.

Количество аргументов присваивается другой переменной: $#(диез). Наконец, имя

процедуры - это $0; переменная $0 не учитывается при подсчете $#.

Сам интерпретатор shell автоматически присваивает значения следующим переменным

(параметрам):

? значение, возвращенное последней командой;

$ номер процесса;

! номер фонового процесса;

# число позиционных параметров, передаваемых в shell;

\* перечень параметров, как одна строка;

@ перечень параметров, как совокупность слов;

- флаги, передаваемые в shell.

При обращении к этим переменным (т.е при использовании их в командном файле - shell-

программе) следует впереди ставить "$".

Пример.

Вызов фала – specific par1 par2 par3 имеющего вид

echo $0 - имя расчета

echo $? - код завершения

echo $$ - идентификатор последнего процесса

echo $! - идентификатор последнего фонового процесса

echo

echo $\* - значения параметров, как строки

echo $@ - значения параметров, как слов

echo

set -au

echo $- - режимы работы интерпретатора

Выдаст на экран

specific - имя расчета

0 - код завершения

499 - идентификатор последнего процесса

98 - идентификатор последнего фонового процесса

par1 par2 par3 - значения параметров, как строки

par1 par2 par3 - значения параметров, как слов

au - режимы работы интерпретатора

Некоторые вспомогательные операторы:

echo - вывод сообщений из текста процедуры на экран.

$ echo "начало строки

> продолжение строки"

# - для обозначения строки комментария в процедуре. (Строка не будет обрабатываться

shell-ом).

banner - вывод сообщения на экран заглавными буквами (например для

идентификации следующих за ним сообщений).

$banner 'hello ira'

HELLO IRA

$

Простейший пример. Здесь оператор echo выполняется в командном режиме.

$shfil p1 pp2 petr

$echo $3

petr

$

Пример

Передача большого числа параметров

echo "$0: Много параметров"

echo " Общее число параметров = $#

Исходное состояние: $1 $5 $9 "

shift

echo "1 сдвиг: первый=$1 пятый=$5 девятый=$9"

shift 2   
 66

echo "1 + 2 = 3 сдвига: первый=$1 пятый=$5 девятый=$9"

perem=`expr $1 + $2 + $3`

echo $perem

Значения параметрам, передаваемым процедуре, можно присваивать и в процессе работы

процедуры с помощью оператора

set - присвоить значения позиционным параметрам;

$set a1 ab2. abc

$echo $1 $2

a1 ab2 - в этом примере параметры указываются в явном виде.

$

Запуск set без параметров выводит список установленных системных переменных:

HOME=/home/sae

PATH=/usr/local/bin:/usr/bin:/bin:.:/usr/bin/X11:

IFS=

LOGNAME=sae

MAIL=/var/spool/mail/sae

PWD=/home/sae/STUDY/SHELL

PS1=${PWD}:" "

PS2=>

SHELL=/bin/bash

TERM=linux

TERMCAP=console|con80x25|dumb|linux:li#25:co#80::

UID=501

Количество позиционных параметров может быть увеличено до необходимого значения

путем "сдвига" их в командной строке влево на одну позицию с помощью команды shift без

аргументов:

shift - сдвинуть позиционные параметры влево на одну позицию

После выполнения shift прежнее значение параметра $1 теряется, значение $1 приобретает

значение $2, значение $2 - значение $3 и т.д..Продолжение предыдущего примера:

$shift

$echo $1 $2

ab2 abc

$

В UNIX при написании операторов важное значение отводится кавычкам

(апострофам):

'...' - для блокирования специальных символов, которые могут быть интерпретированы как

управляющие;

"..." - блокирование наиболее полного набора управляющих символов или указания того,

что здесь будет обрабатываться не сам аргумент, а его значение;

`...` - (обратные кавычки или знак ударения) для указания того, что они обрамляют команду

и здесь будет обрабатываться результат работы этой команды (подстановка результатов

работы указанной команды).

Пример 1.

$ date   
 67

Apr 3 14:27:07 2005

$ set `date`

$ echo $3

14:30:25

$

Пример 2.

$echo `ls`

fil.1

fil.2

...

$echo '`ls`'

# одинарные кавычки блокируют действие обратных кавычек

# т.е. они распечатываются как обычные символы

`ls`

$

Для ввода строки текста со стандартного устройства ввода используется оператор:

read имя1 [имя2 имя3 .] - чтение строки слов со стандартного ввода

Команда вводит строку, состоящую из нескольких полей (слов), со стандартного ввода,

заводит переменную для каждого поля и присваивает первой переменной имя1, второй

переменной - имя2, и т.д. Если имен больше, чем полей в строке, то оставшиеся

переменные будут инициализированы пустым значением. Если полей больше, чем имен

переменных, то последней переменной будет присвоена подстрока введенной строки,

содержащая все оставшиеся поля, включая разделители между ними. В частности, если имя

указано только одно, то соответствующей ему переменной присваивается значение всей

строки целиком.

Пример:

#Текст процедуры:

echo "Введите значения текущих: гг мм ччвв"

read 1v 2v 3v

echo "год 1v"

echo "месяц 2v"

echo "сегодня 3v"

# здесь кавычки используются для блокирования пробелов

#Результат выполнения процедуры:

Введите значения текущих: гг мм ччвв

2005 Maрт 21 9:30 <Enter>

год 2005

месяц Maрт

сегодня 21 9:30

***Задание.*** Создать сценарий реализующий в консольном режиме диалог с пользователем в виде меню. Сценарий должен выполняться циклически пока не выбран пункт «Выход». Первый пункт меню должен выводить информацию о создателе (ФИО, группа) и краткое описание выполняемых действий, второй пункт меню должен вычислять математическое выражение, а остальные пункты реализуют действия указанные в таблице в соответствии с вариантом. Все параметры задаются в результате диалога с пользователем.

Отчет должен содержать краткие теоретические сведения о использованных командах и операторах.

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| **4** | А)Удаление содержимого заданной папки с перемещением всех удаленных объектов в папку tmp.  Б)Проверить существование указанного в параметре файла и выдать сообщение о результате. |

*Листинг сценария:*

i=0

while test $i -ne 5

do

clear

echo 'Меню'

echo '1.Информация'

echo '2.Значение функции'

echo '3.Удаление содержимого заданной папки, с перемещением всех удаленных объектов в папку tmp'

echo '4.Проверить существования указанного в параметре файла и выдать сообщение о результате'

echo '5.Выход'

echo 'Выберите пункт меню'

read i

if test $i -eq 1

then echo 'Автор'

echo 'Валуев Вадим'

echo 'группа ИТП-11'

echo

echo 'О работе'

echo 'Этот сценарий результат лабораторной работы 4'

elif test $i -eq 2

then

echo 'Посчитаем значение функции x=(№Компьютера + №По журналу)\*Возраст'

echo 'Для этого необходимо ввести данные'

echo 'Введите № компьютера'

read nk

echo 'Введите № по журналу'

read nv

echo 'Введите Ваш возраст'

read v

echo 'Значение функции равно'

x=`expr $nk \\* $v + $nv \\* $v`

echo x=$x

elif test $i -eq 3

then

echo 'Введите адрес папки'

read fold

if [ -d $fold ]

then

m= 0

cd $fold

for l in \*

do

echo "$l"

cp $l /tmp

rm $l

m=`expr $m + 1 `

done

echo "Число скопированных файлов: $m"

else echo 'Такой папки не существует'

fi

elif test $i -eq 4

then

if [ -f $1 ]

then echo 'Файл существует'

else echo 'Файл не существует'

fi

elif test $i -eq 5

then echo 'Завершение работы'

fi

read key

done

Результат выполнения работы:

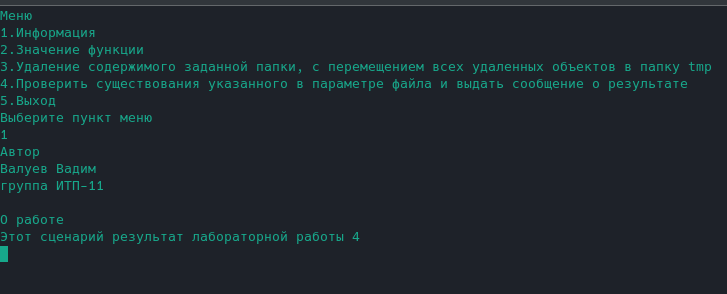


Рисунок 1 - Результат выполнения 1 пункта

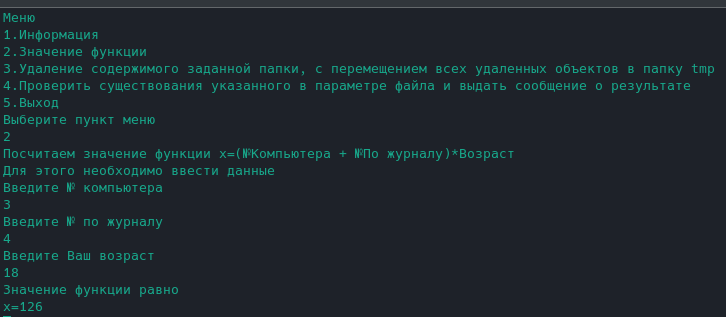


Рисунок 2 - Результат выполнения 2 пункта

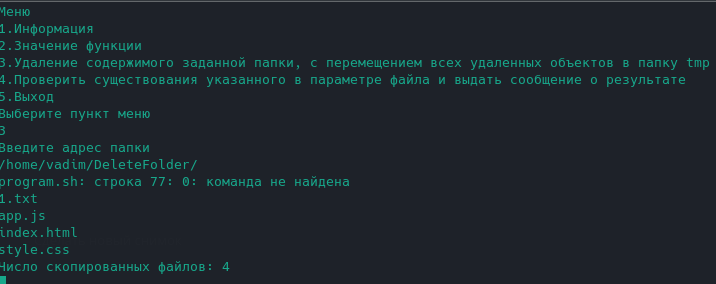


Рисунок 3 - Результат выполнения 3 пункта

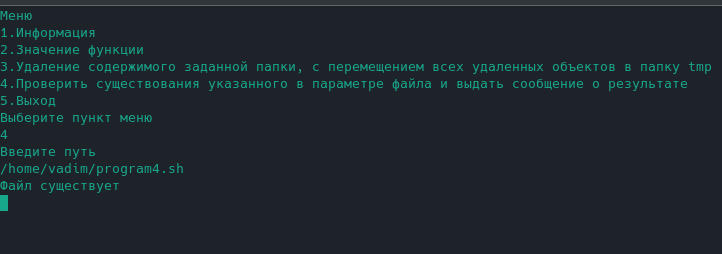


Рисунок 4 - Результат выполнения 4 пункта

***Выводы:*** в ходе выполнения лабораторной работе: изучили синтаксис и операторы для языка Shell, научился экспортировать локальные переменные в среду Shell и управлять ими, получили навыки в построении циклов и ветвлений вычислительных процессов, также была изучена проверка условий.