Отчет по лабораторной работе №13

Дисциплина: Операционные системы

Иванов Сергей Владимирович

Содержание

# 1 Цель работы

Цель данной лабораторной работы - изучить основы программирования в оболочке ОС UNIX, научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 2 Задание

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

* -iinputfile — прочитать данные из указанного файла;
* -ooutputfile — вывести данные в указанный файл;
* -pшаблон — указать шаблон для поиска;
* -C — различать большие и малые буквы;
* -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p.

1. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
2. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).
3. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

# 3 Выполнение лабораторной работы

Создаю файл с разрешением на исполнение (рис. 1).

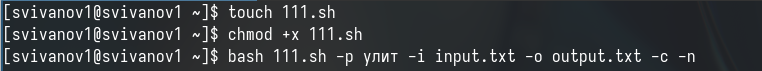


Рис. 1: Создание файла

Командный файл, c командами getopts и grep, который анализирует командную строку с ключами: - -iinputfile — прочитать данные из указанного файла; - -ooutputfile — вывести данные в указанный файл; - -pшаблон — указать шаблон для поиска; - -C — различать большие и малые буквы; - -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p (рис. 2).

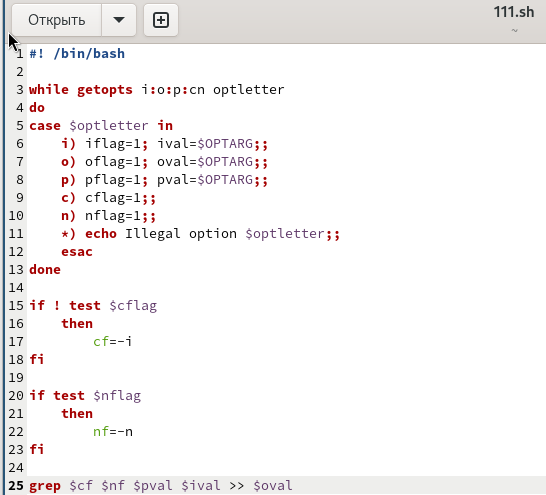


Рис. 2: Код программы

#! /bin/bash  
  
while getopts i:o:p:cn optletter  
do  
case $optletter in  
 i) iflag=1; ival=$OPTARG;;  
 o) oflag=1; oval=$OPTARG;;  
 p) pflag=1; pval=$OPTARG;;  
 c) cflag=1;;  
 n) nflag=1;;  
 \*) echo Illegal option $optletter;;  
 esac  
done  
  
if ! test $cflag  
 then  
 cf=-i  
fi  
  
if test $nflag  
 then  
 nf=-n  
fi  
  
grep $cf $nf $pval $ival >> $oval

Результат работы программы в файле output.txt (рис. 3).

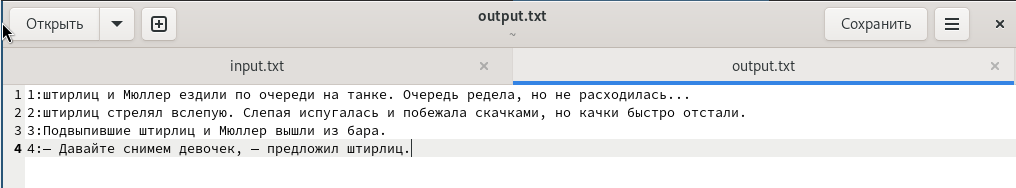


Рис. 3: Результат работы программы

Создаю исполняемый файл для второй программы, также создаю файл 12.сpp для программы на С (рис. 4).

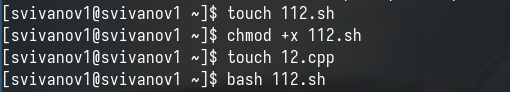


Рис. 4: Создание файла

Пишу программу на языке Си, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку (рис. 5).

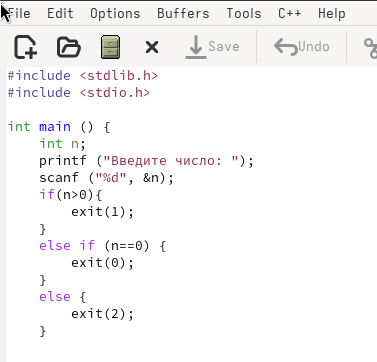


Рис. 5: Код программы на Си

#include <stdlib.h>  
#include <stdio.h>  
  
int main () {  
 int n;  
 printf ("Введите число: ");  
 scanf ("%d", &n);  
 if(n>0){  
 exit(1);  
 }  
 else if (n==0) {  
 exit(0);  
 }  
 else {  
 exit(2);  
 }

Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено (рис. 6).

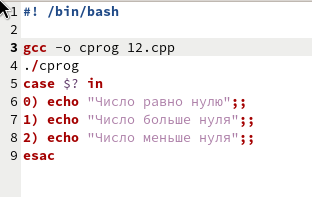


Рис. 6: Код программы

#! /bin/bash  
  
gcc -o cprog 12.c  
./cprog  
case $? in  
0) echo "Число равно нулю";;  
1) echo "Число больше нуля";;  
2) echo "Число меньше нуля";;  
esac

Программа работает корректно (рис. 7).

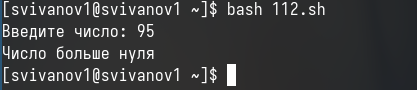


Рис. 7: Результат работы программы

Создаю исполняемый файл для третьей программы (рис. 8).

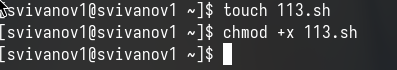


Рис. 8: Создание файла

Командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют) (рис. 9).

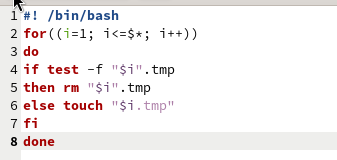


Рис. 9: Код программы

#! /bin/bash  
for((i=1; i<=$\*; i++))  
do  
if test -f "$i".tmp  
then rm "$i".tmp  
else touch "$i.tmp"  
fi  
done

Проверяю, что программа создала файлы и удалила их при соответствующих запросах (рис. 10).

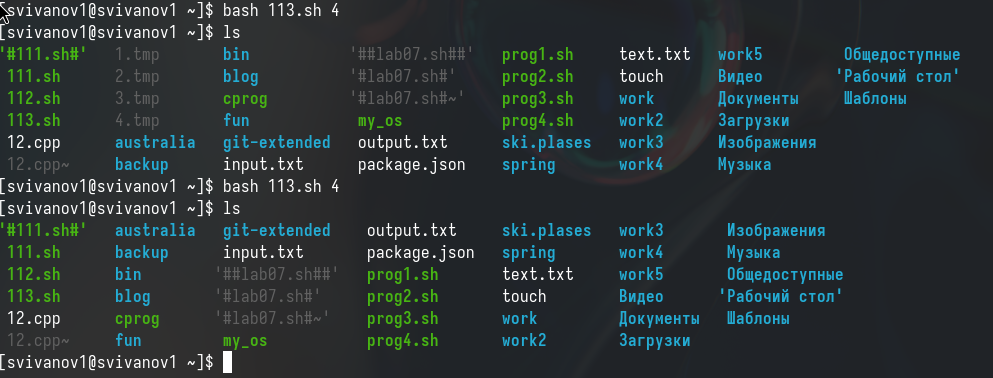


Рис. 10: Результат работы программы

Создаю исполняемый файл для четвертой программы. Это командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find) (рис. 11).

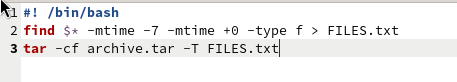


Рис. 11: Код программы

#! /bin/bash  
find $\* -mtime -7 -mtime +0 -type f > FILES.txt  
tar -cf archive.tar -T FILES.txt

Проверяю работу программы (рис. 12).

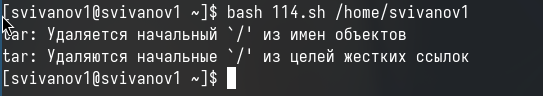


Рис. 12: Результат работы программы

# 4 Вывод

При выполнении данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.