Лабораторная работа №13

Дисциплина: Операционные системы

Жибицкая Евгения Дмитриевна

Содержание

# 1 Цель работы

Изучение основ программирования в оболочке ОС UNIX. Приобритение навыков в написании более сложных командных файлов 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.
2. 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.
3. 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).
4. 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

Начнем выполение задания со 2 пункта. Для этого создадим 2 файла - назовем их 13-2.sh и 13-2.с, запустим редактор emacs для написания самого кода(рис. 1). Не забудем прописать команду chmod +x для наделения файла правом на исполнение.

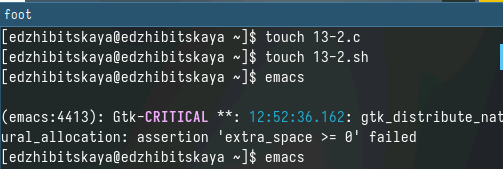


Рис. 1: Создание файлов

Реализуем задание - напишем код на с# и создадим командный файл, анализирующий этот код (рис. 2) (рис. 3).

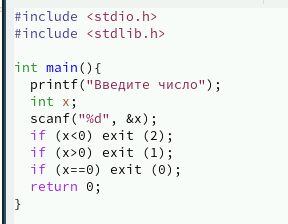


Рис. 2: Код на с#

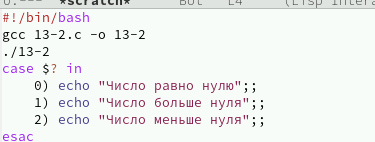


Рис. 3: Программа 2

Программа 2

#!/bin/bash  
gcc 13-2.c -o 13-2  
./13-2  
case $? in  
 0) echo "Число равно нулю";;  
 1) echo "Число больше нуля";;  
 2) echo "Число меньше нуля";;  
esac

Запустим файл и проверим работу программы(рис. 4).

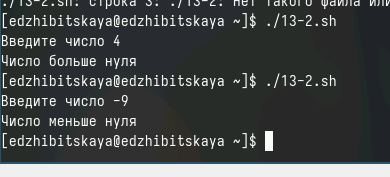


Рис. 4: Запуск 2

Далее перейдем к пункту 1.

Также создадим файл, назовем его 13-1.sh и наделим его правами на исполнение. Далее создадим 2 текстовых файла(в один добавим текст, а второй нужен для записи) и перейдем к написанию программы(рис. 5).

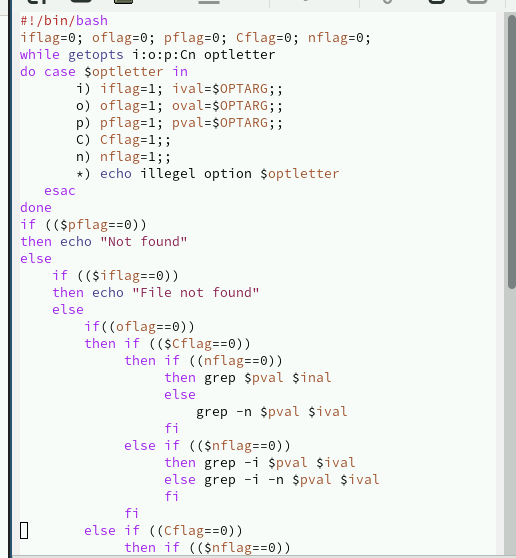


Рис. 5: Программа 1

Программа 1

#!/bin/bash  
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;  
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 illegel option $optletter  
 esac  
done  
if (($pflag==0))  
then echo "Not found"  
else  
 if (($iflag==0))  
 then echo "File not found"  
 else  
 if((oflag==0))  
 then if (($Cflag==0))  
 then if ((nflag==0))  
 then grep $pval $inal  
 else  
 grep -n $pval $ival  
 fi  
 else if (($nflag==0))  
 then grep -i $pval $ival  
 else grep -i -n $pval $ival  
 fi  
 fi  
 else if ((Cflag==0))  
 then if (($nflag==0))  
 then grep $pval $ival > $oval  
 else grep -n $pval $ival > $oval  
 fi  
 else if (($nflag==0))  
 then grep -i $pval $ival > $oval  
 else grep -i -n $pval $ival >$oval  
 fi  
 fi  
 fi  
 fi  
fi

Запустим наш файл и посмотрим на вывод(рис. 6).

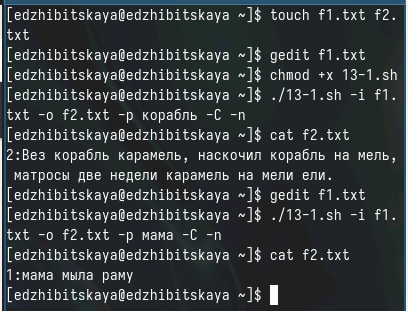


Рис. 6: Запуск 1

Затем создадим файл для 3й программы. Наделим правами доступ и перейдем к написанию кода(рис. 7) и (рис. 8).

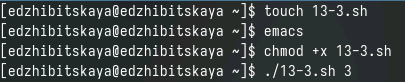


Рис. 7: Создание файла и наделение правами

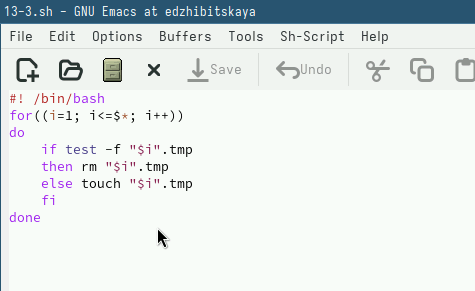


Рис. 8: Программа 3

Программа 3

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

Запустим программу, убедимся, что все работает корректно - файлы создаются и удаляются(рис. 9).

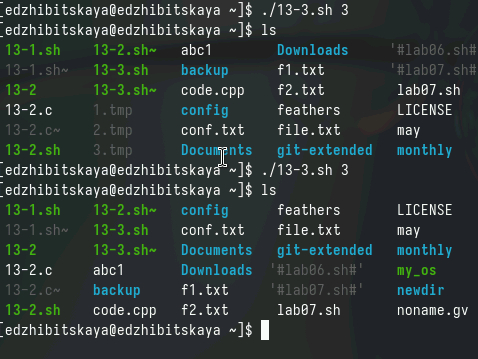


Рис. 9: Запуск 3

Наконец перейдем к 4 пункту. Выполним те же действия, создав и написав программу для исполняемого файла(рис. 10) и (рис. 11).

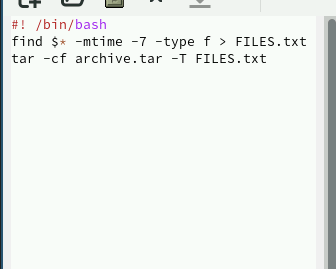


Рис. 10: Программа 4

Программа 4

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

Запуск 4

Рис. 11: Запуск 4

В результате запуска создается архив, с даннными из указанного каталога, причем с изменениями до 7 дней.

# 4 Контрольные вопросы

1. Команда getopts используется для обработки аргументов командной строки в скриптах Shell, чтобы сделать их более гибкими и управляемыми.
2. Метасимволы используются для шаблонного поиска и обработки файлов в командной оболочке UNIX. Они позволяют задавать шаблоны для поиска файлов с определенными именами или расширениями.
3. Операторы управления действиями включают в себя условные операторы (if-then-else), операторы цикла (for, while, until), операторы выбора (case).
4. Для прерывания цикла используются операторы break и continue.
5. Команда false возвращает ложное значение (код ошибки), а команда true - истинное значение (код успеха). Они могут быть использованы для управления потоком выполнения в скриптах.
6. Строка if test -f mani.$s проверяет, существует ли файл с именем, указанным в переменных $s и $i. Опция -f утилиты test указывает на проверку существования обычного файла.
7. Конструкция while выполняет цикл до тех пор, пока условие истинно, а конструкция until выполняет цикл до тех пор, пока условие ложно. Разница заключается в том, что в while условие проверяется перед выполнением тела цикла, а в until - после.

# 5 Выводы

В ходе работы мы изучили основые программирования в оболочке UNIX, написали несколько более сложных командных файлов.