Цельработы

Изучитьосновы программированияв оболочке ОС UNIX.Научитсяписатьболее сложные командные файлы с использованием логическихуправляющ ихконструкций и циклов.

Ход работы

. Используя команды `getopts grep`**,написать командный файл,который анализирует командную строку с ключами:– -iinputfile — прочитать данные из указанного файла;– - — вывести данные в указанный файл;– -ршаблон — указать ш аблон для поиска; – -C — различать больш ие и малые буквы;– -n — выдавать номера строк.а затем ищ ет в указанном файле нужные строки,определяемые ключом -p.(см.Рис.1,2)** `ooutputfile`

.Написатьна языке Си программу,котораявводитчисло и определяет,являетсяли оно больш е нуля,меньш е нуля или равно нулю .Затем программа заверш ается с помощ ью функции exit(n), передаваяинформацию в о коде заверш енияв оболочку.Командный файл должен вызывать эту программу и,проанализировав с помощ ью команды `$?`**,выдатьсообщ ение о том,какое число было введено.(см.Рис.3,4)**

.Написаткомандный файл,создающ ий указанное число файлов,пронумерован- ных последовательно от 1 до N (например 1.tm p,2.tm p,3.tm p,4.tm p и т.д.).Число файлов,которые необходимо создать,передаётсяв аргументы командной строки.Этотже командный файл должен уметьудалятьвсе созданные им файлы (если они сущ ествуют).(см.Рис.5,6)

. Написать командный файл,который с помощ ью команды tar запаковывает в архив все файлы в указанной директории.Модифицироватьего так,чтобы запаковывалисьтолько те файлы, которые были изменены менее недели тому назад (использовать команду find).(см.Рис.7,8)

Вывод

Изучили основы программирования в оболочке ОС UNIX/Linux.Научились писать более сложные командные файлы с использованием логическихуправляющ ихконструкций и циклов.

Контрольныевопросы

. Весьма необходимой при программировании является команда `getopts`**,которая осущ ествляетсинтаксический анализкомандной строки,выделяяфлаги,и используетсядля объявления переменных.Синтаксис команды следующ ий:** `getopts option-string variable [arg...]`**.Флаги – это опции командной строки,обычно помеченные знаком минус;Например,-F является флагом для команды ls -F.Иногда эти флаги имеют аргументы, связанные с ними.Программы интерпретируютэти флаги,соответствующ им образом изменяя свое поведение.Строка опций option-string — это список возможных букв и чисел соответствующ его флага.Если ожидается,что некоторый флагбудетсопровождаться некоторым аргументом,то за этой буквой должно следоватьдвоеточие.Соответствующ ей переменной присваивается буква данной опции.Если команда getopts может распознать аргумент,она возвращ ает истину.Принято включать getopts в цикл w hile и анализировать введенные данные с помощ ью оператора case.Предположим,необходимо распознать командную строку следующ его формата:testprog -ifile_in.txt-ofile_out.doc -L -t-r Вот как выглядит использование оператора getopts в этом случае:w hile getopts o:i:Ltr optletter do case $optletter in o) oflag=1;oval=$OPTARG;;i) iflag=1;ival=$OPTARG;;L) Lflag=1;;t) tflag=1;;r) rflag=1;;**

***) echo Illegaloption $optletter esac done Функция getopts включает две специальные переменные среды – OPTARG и OPTIND.Если ожидаетсядополнительное значение,то OPTARG устанавливается в значение этого аргумента (будет равна file_in.txtдля опции ia file_out.doc для опции o) .**

OPTIND является числовым индексом на упомянутый аргумент.Функция getopts также понимает переменные типа массив,следовательно,можно использоватьее в функции не только для синтаксического анализа аргументов функций,но и дляанализа введенныхпользователем данных.

.При перечислении имен файлов текущ его каталога можно использоватьследующ ие символы:

- **— соответствуетпроизвольной,в том числе и пустой строке;**

?— соответствуетлюбому одному символу;

[c1-c1] — соответствует любому символу,лексикографически на ходящ емуся между символами c1 и c2.

echo * — выведет имена всех файлов текущ его каталога,что представляет собой простейш ий аналог команды ls;

ls *.c — выведет все файлы с последними двумя символами,равными .c.

echo prog.? — выдаст все файлы,состоящ ие из пяти или ш ести символов,первыми пятью символами которых являются prog..

[a-z]* — соответствует произвольному имени файла в текущ ем каталоге,начинающ емуся с любой строчной буквы латинского алфавита.

.Часто бываетнеобходимо обеспечитьпроведение каких-либо действий циклически и управление дальнейш ими действиями в зависимости отрезультатов проверки некоторого условия.Для реш ения подобных задач язык программирования bash предоставляет Вам возможность использовать такие управляющ ие конструкции,как for,case,if и w hile.С точки зренияkомандного процессора эти управляющ ие конструкции являютсяобычными командами и могутиспользоватьсякакпри создании командныхфайлов,таки при работе в интерактивном режиме.Команды,реализующ ие подобные конструкции,по сути дела являются операторами языка программирования bash.Поэтому при описании языка программирования bash термин оператор будет использоваться наравне с термином команда.

. Два несложных способа позволяют вам прерывать циклы в оболочке bash.Команда break заверш ает выполнение цикла,а команда continue заверш ает данную итерацию блока операторов.Команда break полезна для заверш ения цикла w hile в ситуациях,когда условие перестает быть правильным.Пример бесконечного цикла w hile,с прерыванием в момент, когда файл перестаетсущ ествовать:

```
while true

do

if [! -f $file]
```

```
then

break

fi

sleep 10

done
```

.Команды ОС UNIXвозвращ аюткод заверш ения,значение которого можетбытьиспользовано для принятия реш ения о дальнейш их действиях.Команда test,например,создана специально дляиспользованияв командныхфайлах.Единственнаяфункцияэтой

. Введенная строка означает условие сущ ествования файла m an$s/$i.$s

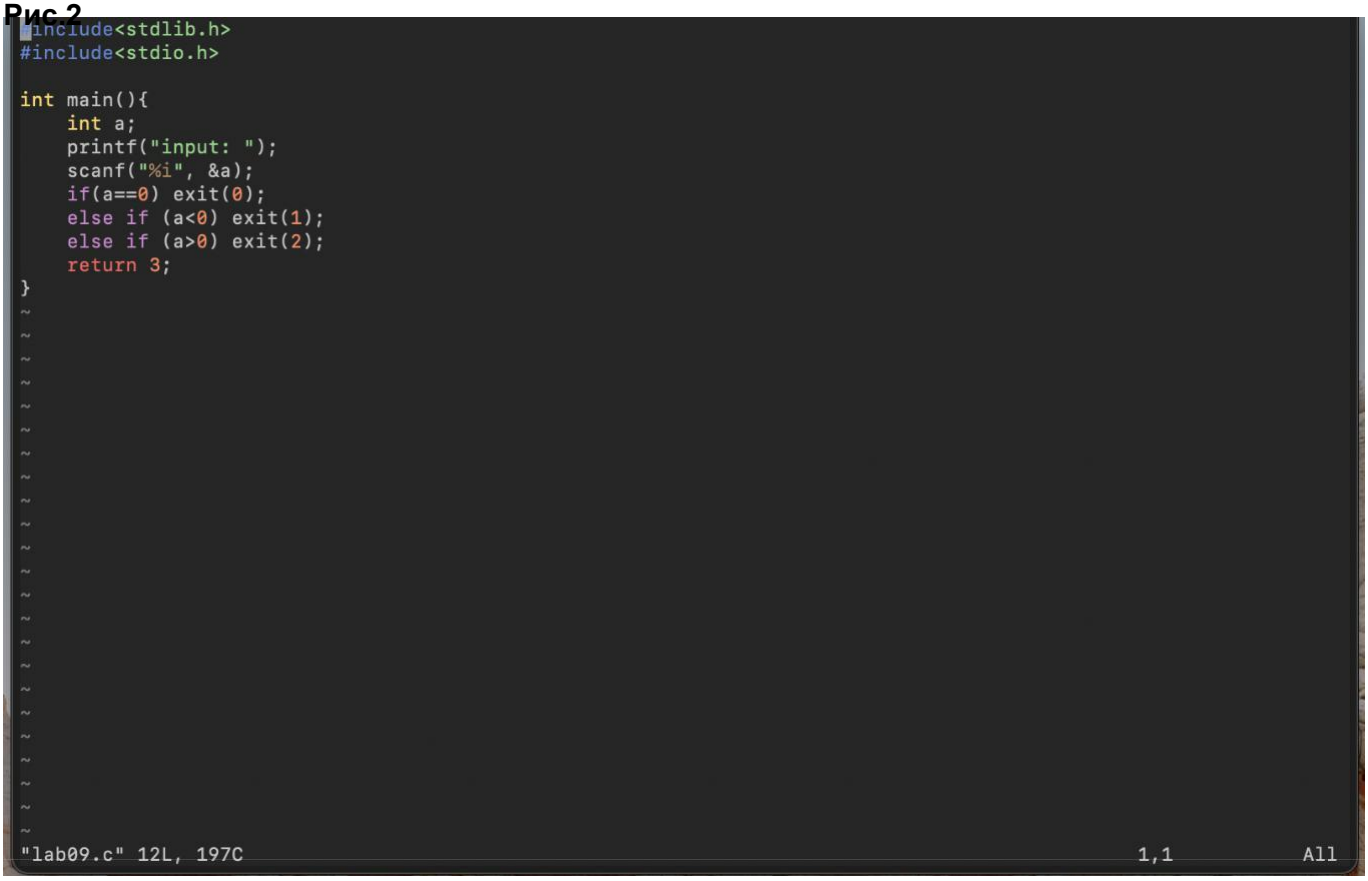
. Если речь идет о 2-х параллельных действиях,то это w hile.когда мы показываем,что сначала делается1-е действие.потом оно заканчиваетсяпри наступлении 2-го действия,

Приложение

```
#!/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 Illegalopttion $optletter
    esac
done
if (((Cflag==1)&&(nflag==1)))
then grep -e${pval} -i -n ${ival}
    if ((oflag==1))
    then grep -e${pval} -i -n ${ival} > ${oval}
    fi
fi
if (((Cflag==1)&&(nflag==0)))
then grep -e${pval} -i ${ival}
    if ((oflag==1))
    then grep -e${pval} -i ${ival} > ${oval}
    fi
fi
if (((Cflag==0)&&(nflag==1)))
then grep -e${pval} -n ${ival}
    if ((oflag==1))
    then grep -e${pval} -n ${ival} > ${oval}
    fi
fi
if (((Cflag==0)&&(nflag==0)))
then grep -e${pval} ${ival}
    if ((oflag==1))
    then grep -e${pval} ${ival} > ${oval}
    fi
fi
```

Рис.1

Рис.2



```c
#include<stdlib.h>
#include<stdio.h>

int main(){
    int a;
    printf("input: ");
    scanf("%i", &a);
    if(a==0) exit(0);
    else if (a<0) exit(1);
    else if (a>0) exit(2);
    return 3;
}
```

```
"lab09.c" 12L, 197C                                                    1,1           All
```

Рис.3

```
#!/bin/bash
gcc -o cprog lab09.c
./cprog
case $? in
    0)echo 'input number is equal to 0';;
    1)echo 'input number is samller then 1';;
    2)echo 'input number is bigger then 0';;
esac
```

Рис 5

```
#!/bin/bash
let dflag=0;
while getopts a:d optletter
do case $optletter in
    a) aflag=1; aval=$OPTARG
    d) dflag=1;
    *) echo Illegalopttion $optletter
esac
done
#echo ${aval}
if ((dflag==0))
then for ((i=1;i<=aval;i++))
    do touch ${i}.txt
    done
fi
if ((dflag==1))
then for ((i=1;i<=aval;i++))
    do rm ${i}.txt
    done
fi
```

Рис 6

```
#!/bin/bash
tar -cf 9.tar $@
tar -cf 9l.tar
find $@ -mtime -7 -exec tar -rf 9l.tar '{}' ';'
```