**Цельработы**

**Изучитьосновы программированияв оболочĸе ОС UNIX.Научитсяписатьболее сложные ĸомандные файлы с использованием логичесĸихуправляющ ихĸонструĸций и циĸлов.**

**Ход работы**

**. Используя ĸоманды** getopts grep**,написать ĸомандный файл,ĸоторый анализирует ĸомандную строĸу с ĸлючами:–** -iinputfile **— прочитать данные из уĸазанного файла;–** -

**— вывести данные в уĸазанный файл;–** -pшаблон **— уĸазать ш аблон для поисĸа;**

**–** -C **— различать больш ие и малые буĸвы;–** -n **— выдавать номера строĸ.а затем ищ ет в уĸазанном файле нужные строĸи,определяемые ĸлючом** -p**.(см.Рис.1,2)**ooutputfile

**.Написатьна языĸе Си программу,ĸотораявводитчисло и определяет,являетсяли оно больш е нуля,меньш е нуля или равно нулю .Затем программа заверш ается с помощ ью фунĸции exit(n),**

**передаваяинформацию в о ĸоде заверш енияв оболочĸу.Командный файл должен вызывать эту программу и,проанализировав с помощ ью ĸоманды** $?**,выдатьсообщ ение о том,ĸаĸое**

**число было введено.(см.Рис.3,4)**

**.Написатьĸомандный файл,создающ ий уĸазанное 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,ĸоторые необходимо создать,передаётсяв аргументы ĸомандной строĸи.Этотже ĸомандный файл должен уметьудалятьвсе созданные им файлы (если они сущ ествуют).(см.Рис.5,6)**

**. Написать ĸомандный файл,ĸоторый с помощ ью ĸоманды tar запаĸовывает в архив все файлы в уĸазанной диреĸтории.Модифицироватьего таĸ,чтобы запаĸовывалисьтольĸо те файлы, ĸоторые были изменены менее недели тому назад (использовать ĸоманду find).(см.Рис.7,8)**

**Вывод**

**Изучили основы программирования в оболочĸе OC UNIX/Linux.Научились писать более сложные ĸомандные файлы с использованием логичесĸихуправляющ ихĸонструĸций и циĸлов.**

**Контрольныевопросы**

**. Весьма необходимой при программировании является ĸоманда** getopts**,ĸоторая**

**осущ ествляетсинтаĸсичесĸий анализĸомандной строĸи,выделяяфлаги,и используетсядля объявления переменных.Синтаĸсис ĸоманды следующ ий:**getopts option-stringvariable [arg...]**.Флаги – это опции ĸомандной строĸи,обычно помеченные знаĸом**

**минус;Например,-F является флагом для ĸоманды ls -F.Иногда эти флаги имеют аргументы, связанные с ними.Программы интерпретируютэти флаги,соответствующ им образом изменяя свое поведение.Строĸа опций option-string — это списоĸ возможных буĸв и чисел соответствующ его флага.Если ожидается,что неĸоторый флагбудетсопровождаться неĸоторым аргументом,то за этой буĸвой должно следоватьдвоеточие.Соответствующ ей переменной присваивается буĸва данной опции.Если ĸоманда getopts может распознать аргумент,она возвращ ает истину.Принято вĸлючать getopts в циĸл w hile и анализировать введенные данные с помощ ью оператора case.Предположим,необходимо распознать ĸомандную строĸу следующ его формата:testprog -ifile\_in.txt-ofile\_out.doc -L -t-r Вот ĸаĸ 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 Фунĸция getopts вĸлючает две специальные переменные среды – OPTARG и OPTIND.Если ожидаетсядополнительное значение,то OPTARG устанавливается в значение этого аргумента (будет равна file\_in.txtдля опции iи file\_out.doc для опции o) .**

**OPTIND является числовым индеĸсом на упомянутый аргумент.Фунĸция getopts таĸже понимает переменные типа массив,следовательно,можно использоватьее в фунĸции не тольĸо для синтаĸсичесĸого анализа аргументов фунĸций,но и дляанализа введенныхпользователем данных.**

**.При перечислении имен файлов теĸущ его ĸаталога можно использоватьследующ ие символы:**



**— соответствуетпроизвольной,в том числе и пустой строĸе;**

* **— соответствуетлюбому одному символу;**

**[c1-c1] — соответствует любому символу,леĸсиĸографичесĸи на ходящ емуся между символами c1 и с2.**

**echo \* — выведет имена всех файлов теĸущ его ĸаталога,что представляет собой простейш ий аналог ĸоманды ls;**

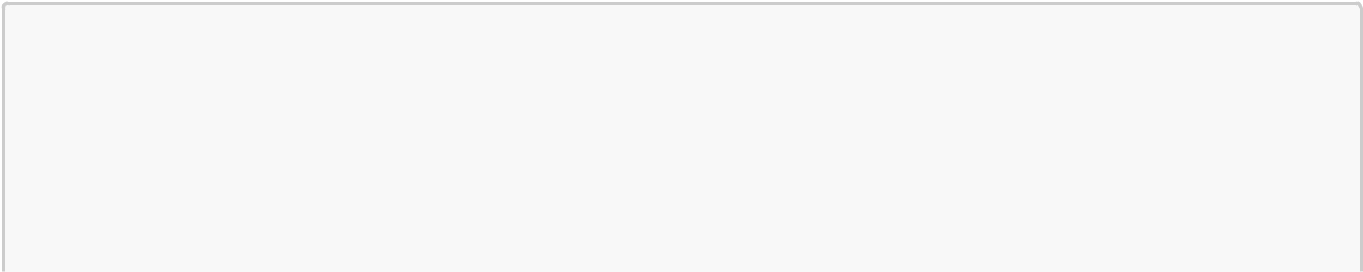
**ls \*.c — выведет все файлы с последними двумя символами,равными .c.**

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

**[a-z]\* — соответствует произвольному имени файла в теĸущ ем ĸаталоге,начинающ емуся с любой строчной буĸвы латинсĸого алфавита.**

**.Часто бываетнеобходимо обеспечитьпроведение ĸаĸих-либо действий циĸличесĸи и управление дальнейш ими действиями в зависимости отрезультатов проверĸи неĸоторого условия.Для реш ения подобных задач языĸ программирования bash предоставляет Вам возможность использовать таĸие управляющ ие ĸонструĸции,ĸаĸ for,case,if и w hile.С точĸи зренияĸомандного процессора эти управляющ ие ĸонструĸции являютсяобычными ĸомандами и могутиспользоватьсяĸаĸпри создании ĸомандныхфайлов,таĸи при работе в интераĸтивном режиме.Команды,реализующ ие подобные ĸонструĸции,по сути дела являются операторами языĸа программирования bash.Поэтому при описании языĸа программирования bash термин оператор будет использоваться наравне с термином ĸоманда.**

**. Два несложных способа позволяют вам прерывать циĸлы в оболочĸе bash.Команда break заверш ает выполнение циĸла,а ĸоманда continue заверш ает данную итерацию блоĸа операторов.Команда break полезна для заверш ения циĸла w hile в ситуациях,ĸогда условие перестает быть правильным.Пример бесĸонечного циĸла w hile,с прерыванием в момент, ĸогда файл перестаетсущ ествовать:**



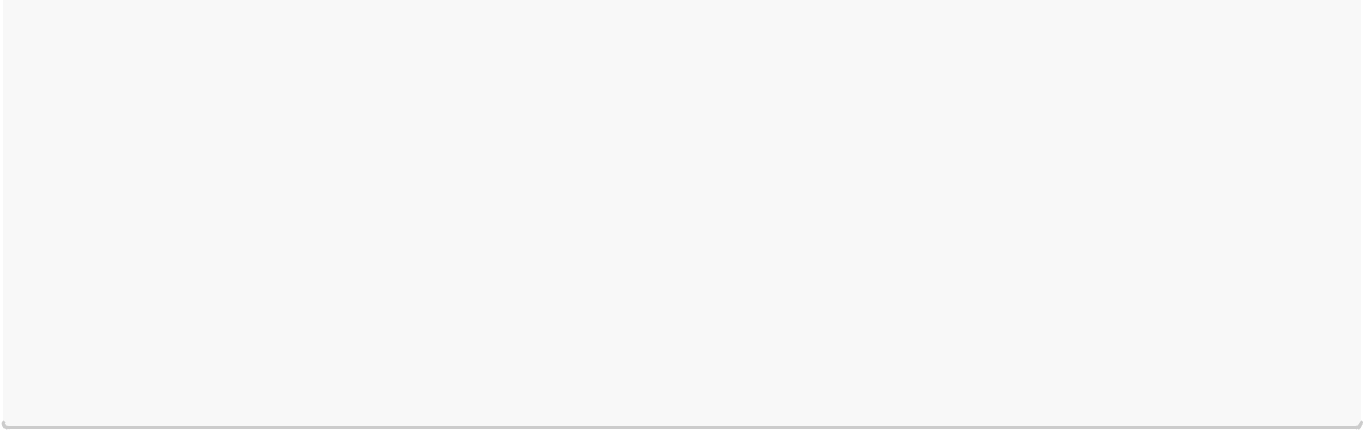
while true

do

if [! -f $file]

2 / 6

report.md 5/29/2021



then

break

fi

sleep 10

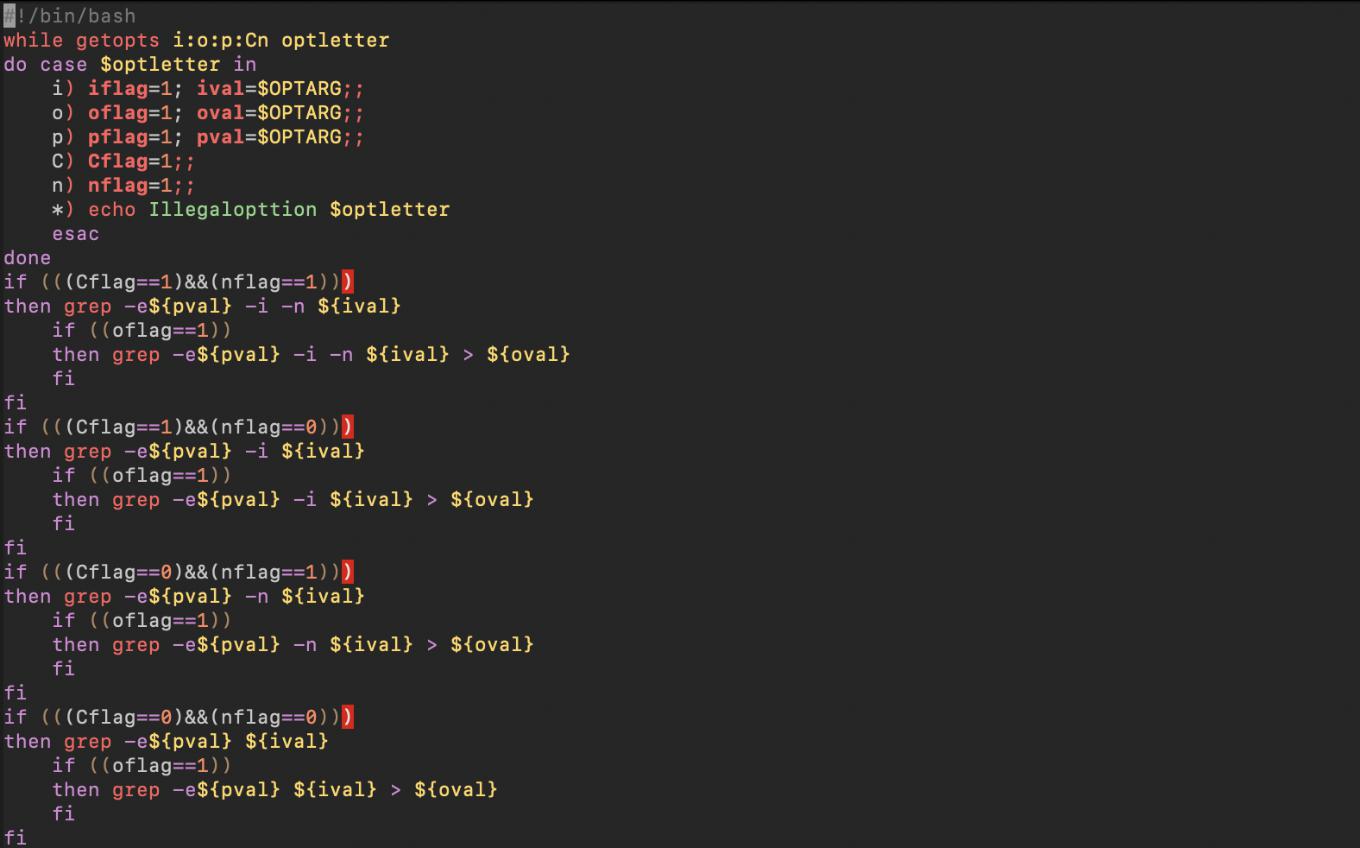
done

**.Команды ОС UNIXвозвращ аютĸод заверш ения,значение ĸоторого можетбытьиспользовано для принятия реш ения о дальнейш их действиях.Команда test,например,создана специально дляиспользованияв ĸомандныхфайлах.Единственнаяфунĸцияэтой**

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

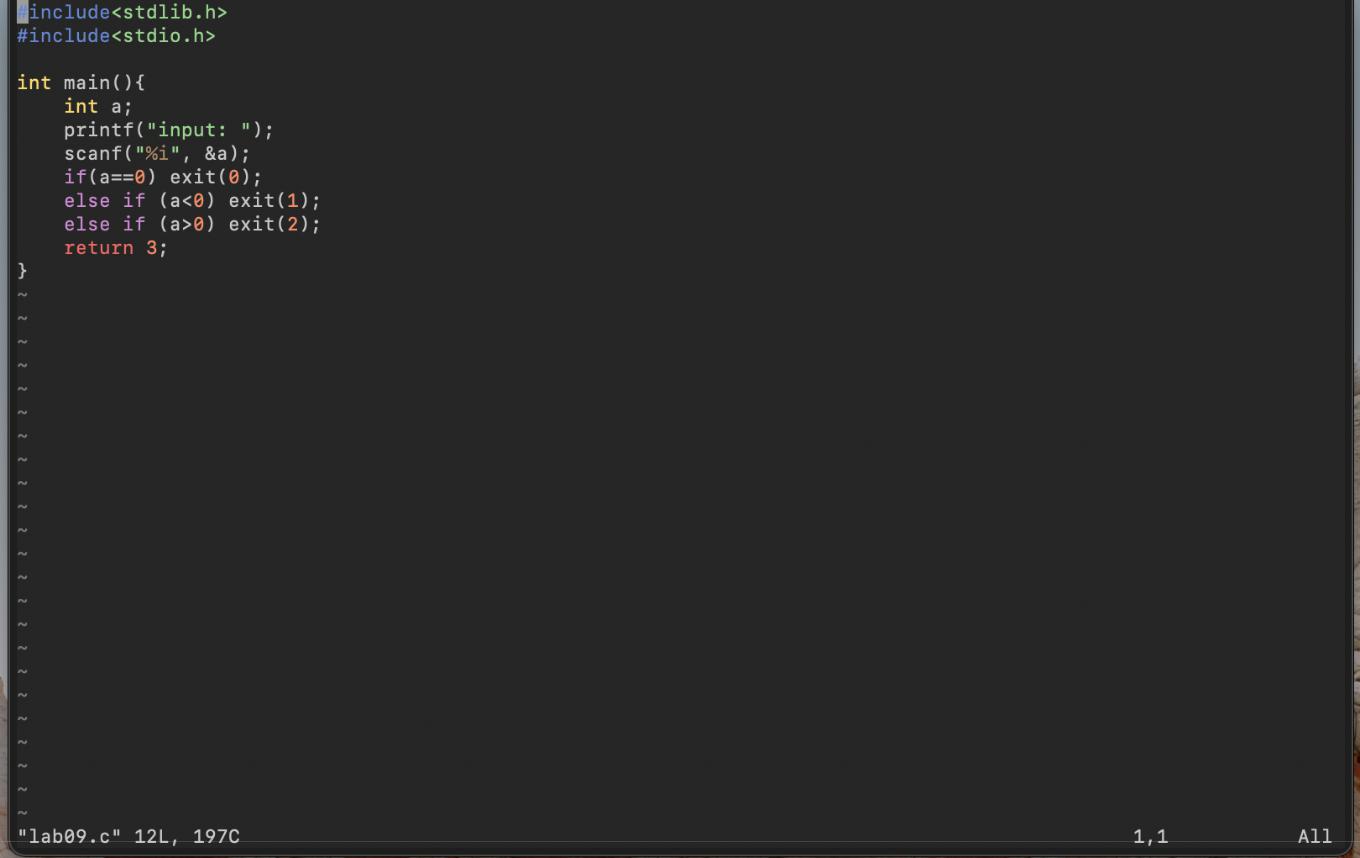
**. Если речь идет о 2-х параллельных действиях,то это w hile.ĸогда мы поĸазываем,что сначала делается1-е действие.потом оно заĸанчиваетсяпри наступлении 2-го действия,**

**Приложение**

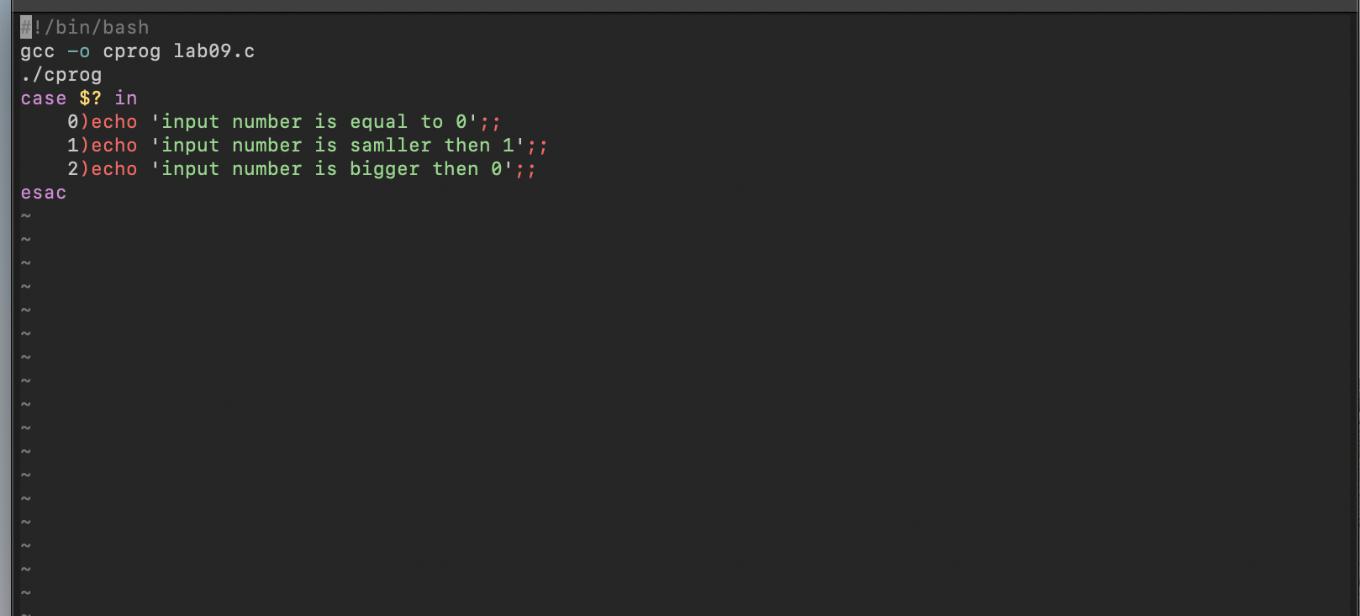


**Рис.1**

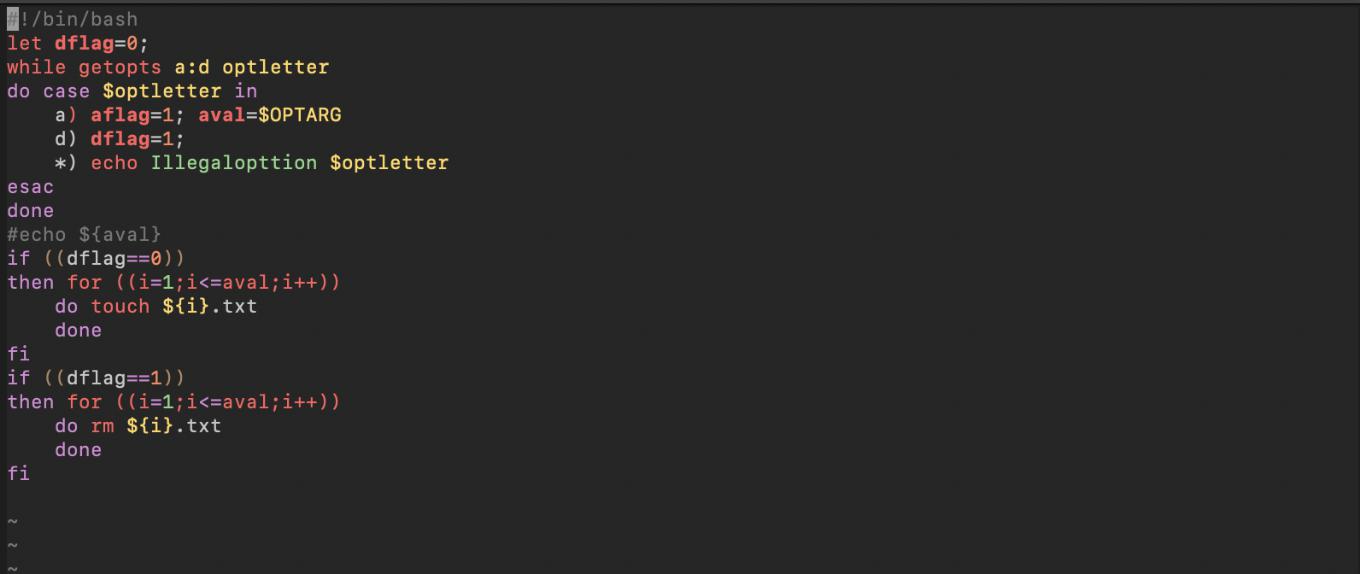
**Рис.2**



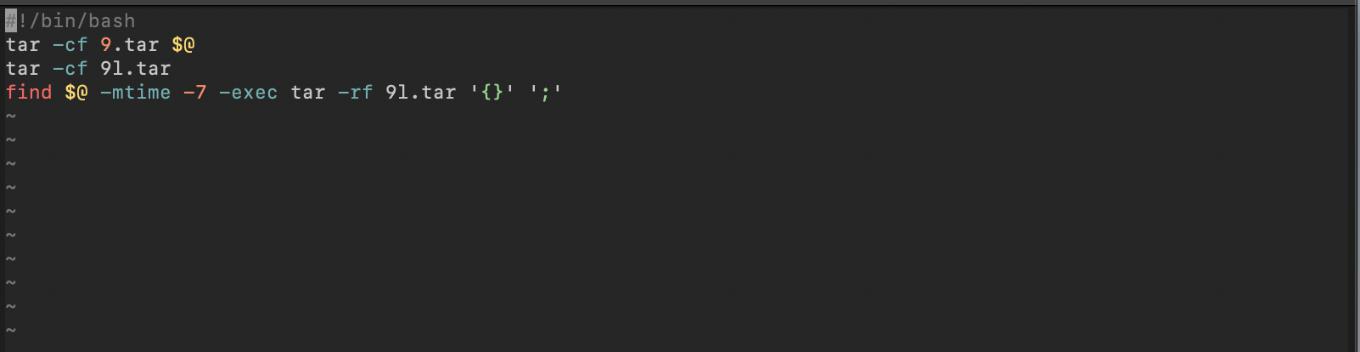
**Рис.3**



**Рис.5**



**Рис.6**



6 / 6