

Отчет по лабораторной работе №13

Простейший вариант

Кривобоков Юрий Дмитриевич

Содержание

1	Цель	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Контрольные вопросы	12
6	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	1	9
4.2	2	10
4.3	3	10
4.4	4	11

Список таблиц

1 Цель

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: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — вывести данные в указанный файл; `-r`шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.
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 `o` 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`).

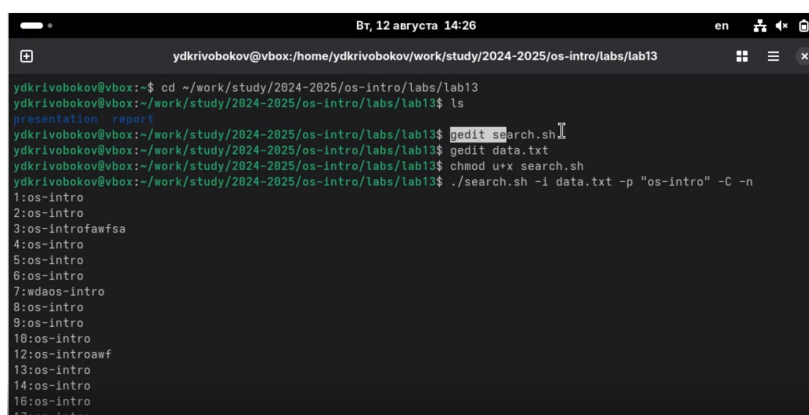
3 Теоретическое введение

Bash (Bourne Again Shell) — это мощная командная оболочка Unix, которая используется для выполнения различных задач в терминале. Bash предоставляет интерактивный интерфейс, в котором пользователи могут вводить команды, а затем получать результаты. Она также поддерживает скрипты оболочки, которые представляют собой текстовые файлы, содержащие последовательность команд Bash для автоматизации задач. Bash широко используется в средах Unix и Linux, а также поддерживается Windows с помощью подсистемы Windows для Linux (WSL). Перечислим основные возможности этой оболочки. Обработка команд. Bash может обрабатывать как простые, так и сложные команды. Простые состоят из одного действия и, возможно, некоторых аргументов. Сложные команды могут содержать несколько простых, объединенных с помощью операторов конвейера (`|`), перенаправления ввода и вывода (`<`, `>`, `>>`), условных операторов (`if/else`, `case/esac`, `while/do`). О них мы расскажем позже. Расширенный ввод, редактирование строк. Оболочка предоставляет функции расширенного ввода, такие как автодополнение, которое предлагает возможные варианты завершения команд и имен файлов по мере их ввода. Она также поддерживает историю, позволяя пользователям просматривать ранее введенные команды, а затем повторно их использовать. Возможность создания и запуска скриптов. Скрипты оболочки — это текстовые файлы, содержащие последовательность команд. Их можно создавать с помощью текстового редактора, а затем запускать в терминале, что дает возможность пользователям автоматизи-

ровать задачи и управлять системой. Скрипты могут содержать условные операторы, циклы, функции для обеспечения дополнительной гибкости и контроля.

4 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — вывести данные в указанный файл; `-r`шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.(рис. **fig:001**).



```
ydkrivobokov@vbox:~/home/ydkrivobokov/work/study/2024-2025/os-intro/labs/lab13$ cd ~/work/study/2024-2025/os-intro/labs/lab13
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ ls
presentation  report
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ edit search.sh
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ gedit data.txt
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ chmod u+x search.sh
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ ./search.sh -i data.txt -p "os-intro" -C -n
1:os-intro
2:os-intro
3:os-introafwfsa
4:os-intro
5:os-intro
6:os-intro
7:wdaos-intro
8:os-intro
9:os-intro
10:os-intro
12:os-introawf
13:os-intro
14:os-intro
16:os-intro
17:os-intro
```

Рисунок 4.1: 1

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. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено. (рис. **fig:002**).

```

15 # p - требует аргумент (pattern)
16 # c - без аргумента (включает чувствительность к регистру)
17 # n - без аргумента (включает вывод номеров строк)
18 while getopts ":i:o:p:Cn" opt; do
19     case $opt in
20         i) inputfile="$OPTARG" ;;           # Сохраняем аргумент после -i в inputfile
21         o) outputfile="$OPTARG" ;;         # Сохраняем аргумент после -o в outputfile
22         p) pattern="$OPTARG" ;;             # Сохраняем аргумент после -p в pattern
23         C) case_sensitive=1 ;;              # Включаем чувствительность к регистру
24         n) show_numbers=1 ;;                # Включаем вывод номеров строк
25         \?) echo "Неверный ключ: -$OPTARG" >&2; exit 1 ;; # Ошибка: неизвестный ключ
26         :) echo "Ключ -$OPTARG требует аргумент." >&2; exit 1 ;; # Ошибка: пропущен аргумент
27     esac
28 done
29
30 # Проверка обязательных параметров
31 # Если inputfile или pattern не указаны, выводим подсказку и завершаем скрипт
32 if [ -z "$inputfile" ] || [ -z "$pattern" ]; then
33     echo "Использование: $0 -i inputfile -p pattern [-o outputfile] [-C] [-n]"
34     exit 1
35 fi

```

Рисунок 4.2: 2

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, передается в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). (рис. fig:003).

```

1 os-intawdasdro
2 os-inawdasdro
3 os-introafwfsa
4 os-inwadatro
5 os-intrawdawao
6 os-insdawtro
7 wdaos-intro
8 os-intro
9 os-intro
10 os-intro
11 os-sadaintro
12 os-introawf
13 os-intro
14 os-intro
15 os-intrasdawdo
16 os-intro
17 os-intro
18 os-intro
19 os-introasfawf
20 os-intro
21 os-introos-intro

```

Рисунок 4.3: 3

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). (рис. fig:004).

```
21:05 ~$ cat os-intro
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ ./search.sh -i data.txt -p "os-intro" -o os-intro.txt
Результаты сохранены в os-intro.txt
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ gedit search.sh
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ gedit data.txt
ydkrivobokov@vbox:~/work/study/2024-2025/os-intro/labs/lab13$ ./search.sh -i data.txt -p "os-intro" -o o.
```

Рисунок 4.4: 4

5 Контрольные вопросы

1. Каково предназначение команды `getopts`? Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable`. 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 интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае:

```
while
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 Illegal
option $optletter esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента

(будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`) . `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. Какое отношение метасимволы имеют к генерации имён файлов? При перечислении имён файлов текущего каталога можно использовать следующие символы: `*` – соответствует произвольной, в том числе и пустой строке; `?` – соответствует любому одинарному символу; `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` – выведет все файлы с последними двумя символами, совпадающими с `.c`. `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `[a-z]` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.
3. Какие операторы управления действиями вы знаете? Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами

языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4. Какие операторы используются для прерывания цикла? Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.
5. Для чего нужны команды `false` и `true`? Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т.е. ложь).
6. Что означает строка `if test -f mans/i.s, ?if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
7. Объясните различия между конструкциями `while` и `until`. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд

в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

6 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы