

# **Отчёт к лабораторной работе №14**

**Программирование в командном процессоре ОС UNIX. Расширенное  
программирование**

Четвергова Мария Викторовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Задание 1 . . . . .	7
3.2	Задание 2 . . . . .	8
3.3	Задание 3 . . . . .	10
<b>4</b>	<b>Выводы</b>	<b>12</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>13</b>

## Список иллюстраций

3.1	Создание и настройка доступа необходимых файлов для задания	7
3.2	Создание и настройка доступа необходимых файлов для задания	9
3.3	Скрипт программы для задания 2 . . . . .	9
3.4	Работа программы . . . . .	10
3.5	Создание и настройка доступа необходимых файлов для задания	10
3.6	Скрипт программы для задания 3 . . . . .	10
3.7	Работа программы . . . . .	11

## Список таблиц

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

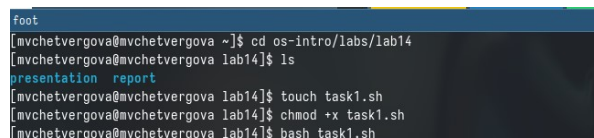
## 2 Задание

Выполнить 3 задания, направленных на создание скриптовых файлов на языке bash.

## 3 Выполнение лабораторной работы

### 3.1 Задание 1

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой ( $> /dev/tty\#$ , где  $\#$  — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.



```
foot
[mvchetvergova@mvchetvergova ~]$ cd os-intro/labs/lab14
[mvchetvergova@mvchetvergova lab14]$ ls
presentation  report
[mvchetvergova@mvchetvergova lab14]$ touch task1.sh
[mvchetvergova@mvchetvergova lab14]$ chmod +x task1.sh
[mvchetvergova@mvchetvergova lab14]$ bash task1.sh
```

Рис. 3.1: Создание и настройка доступа необходимых файлов для задания







```
fast
330 [4mL530[24m(1)
330 [4mL530[24m(1)
330 [1mNAME330[0m
ls - list directory contents
330 [1mSYNOPSIS330[0m
330 [1mL530[22m[ESC[4mOPTIO330[24m]... [ESC[4mFILE330[24m]...
330 [1mDESCRIPTION330[0m
List information about the files (the current directory by default). Sort entries alphabetically if none
of 330 [1m-cftuvSU330[22m or 330 [1m--sort 330[22m is specified.
Mandatory arguments to long options are mandatory for short options too.
330 [1m-s330[22m, 330 [1m-a330[22m
do not ignore entries starting with .
330 [1m-A330[22m, 330 [1m--almost-all330[22m
do not list implied . and ..
330 [1m-author330[22m
with 330 [1m-l330[22m, print the author of each file
330 [1m-l330[22m, 330 [1m--escape330[22m
print C-style escapes for nongraphical characters
330 [1m--block-size=330[22m[ESC[4mSIZE330[24m]
with 330 [1m-l330[22m, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format b
elow
330 [1m-B330[22m, 330 [1m--ignore-backup330[22m
do not list implied entries ending with ~
330 [1m-e 330[22m with 330 [1m-l330[22m: sort by, and show, ctime (time of last change of file status inform
ation); with 330 [1m-l330[22m: show
ctime and sort by name; otherwise: sort by ctime, newest first
330 [1m-C 330[22m list entries by columns
```

Рис. 3.4: Работа программы

### 3.3 Задание 3

- Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинско-го алфавита. Учтите, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

```
[mvchetvergova@mvchetvergova lab14]$ nano task2.sh
[mvchetvergova@mvchetvergova lab14]$ touch task3.sh
[mvchetvergova@mvchetvergova lab14]$ chmod +x task2.sh
[mvchetvergova@mvchetvergova lab14]$
```

Рис. 3.5: Создание и настройка доступа необходимых файлов для задания

```
mvchetvergova [PaSonae] - Oracle VM VirtualBox
fast
GNU nano 7.2 task3.sh
#!/bin/bash
a=$1
for ((i=0;i<$1;i++))
do
  ((char=$RANDOM%26+1))
  case $char in
    1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;;
    6) echo -n f;; 7) echo -n g;; 8) echo -n h;; 9) echo -n i;; 10) echo -n j;;
    11) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;;
    16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;;
    21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
  esac
done
echo
```

Рис. 3.6: Скрипт программы для задания 3

```
[evchetergova@evchetergova lab14]$ chmod +x task3.sh
[evchetergova@evchetergova lab14]$ nano task3.sh
[evchetergova@evchetergova lab14]$ bash task3.sh 4
1knu
[evchetergova@evchetergova lab14]$ bash task3.sh 10
y1hfaevvec
[evchetergova@evchetergova lab14]$ bash task3.sh 132
htmsxezhvydfnuirgvsedprqryuqadavpzw1qi1tahgz11taihrubwy1vcifnwgwnpggdzhoswsnmbdaqzfvzvouuogdzd1jggpc1arntfvdajh
vksaqgbnkookw
[evchetergova@evchetergova lab14]$
```

Рис. 3.7: Работа программы

## 4 Выводы

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

## 5 Ответы на контрольные вопросы

Найдите синтаксическую ошибку в следующей строке: `1 while [$1 != "exit"]` В данной строчке допущены следующие ошибки: не хватает пробелов после первой скобки [ и перед второй скобкой ] выражение `$1` необходимо взять в `"`, потому что эта переменная может содержать пробелы Таким образом, правильный вариант должен выглядеть так: `while [ "$1" != "exit" ]`

Как объединить (конкатенация) несколько строк в одну? Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: `VAR1="Hello," VAR2=" World" VAR3="$VAR1$VAR2" echo "$VAR3"` : *Hello, World* : `VAR1 = "Hello,"VAR1+ = "World"echo"VAR1"`  
Результат: *Hello, World*

Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? Команда `seq` в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: `seq LAST`: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение `is` не выдает. `seq FIRST LAST`: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. `seq FIRST INCREMENT LAST`: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT. Если LAST меньше, чем FIRST, он не производит вывод. `seq -f «FORMAT» FIRST INCREMENT LAST`: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются

необязательными. `seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО`: Эта команда используется для `STRING` для разделения чисел. По умолчанию это значение равно `/n`. `FIRST` и `INCREMENT` являются необязательными. `seq -w FIRST INCREMENT LAST`: эта команда используется для выравнивания ширины путем заполнения начальными нулями. `FIRST` и `INCREMENT` являются необязательными.

Какой результат даст вычисление выражения `$((10/3))`? Результатом данного выражения `$((10/3))` будет `3`, потому что это целочисленное деление без остатка.

Укажите кратко основные отличия командной оболочки `zsh` от `bash`.  
Отличия командной оболочки `zsh` от `bash`: В `zsh` более быстрое автодополнение для `cd` с помощью `Tab` В `zsh` существует калькулятор `zcalc`, способный выполнять вычисления внутри терминала В `zsh` поддерживаются числа с плавающей запятой В `zsh` поддерживаются структуры данных «хэш» В `zsh` поддерживается раскрытие полного пути на основе неполных данных В `zsh` поддерживается замена части пути В `zsh` есть возможность отображать разделенный экран, такой же как разделенный экран `vim`

Проверьте, верен ли синтаксис данной конструкции `1 for ((a=1; a <= LIMIT; a++)) for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать `$` перед переменными `()`.

Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? Преимущества и недостатки скриптового языка `bash`:

Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS Удобное перенаправление ввода/вывода Большое количество команд для работы с файловыми системами Linux Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка `bash`: Дополнительные библиотеки других языков позволяют выполнить больше действий Bash не является языком общего назначения Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь,

отражаются на скорости выполнения этого скрипта Скрипты, написанные на `bash`, нельзя запустить на других операционных системах без дополнительных действий.