

# **Отчёт по лабораторной работе №13**

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

Николаев Дмитрий Иванович

# Содержание

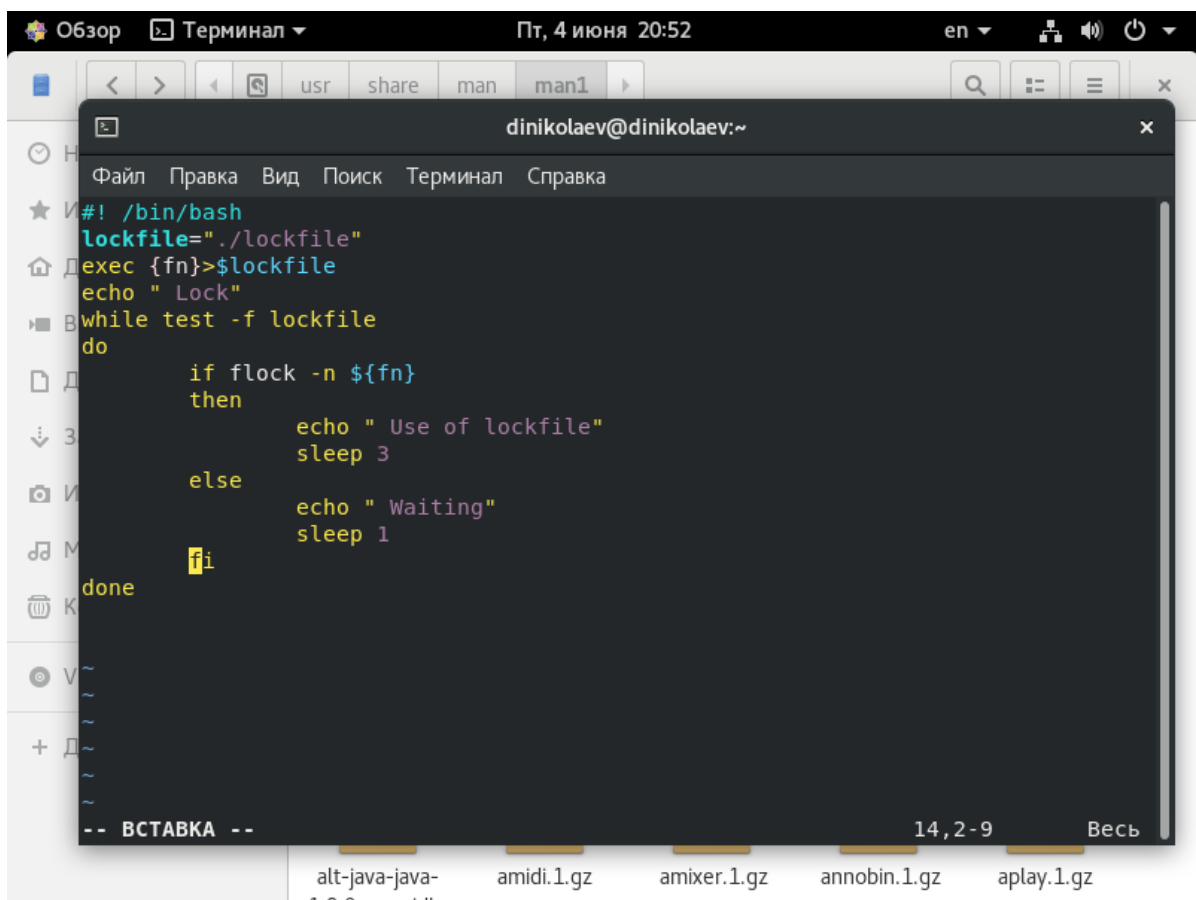
<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
2.1	Контрольные вопросы . . . . .	10
<b>3</b>	<b>Выводы</b>	<b>12</b>

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

- 1) Написал командный файл, который реализует упрощённый механизм семафоров для доступа к необходимому ресурсу.

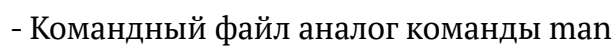


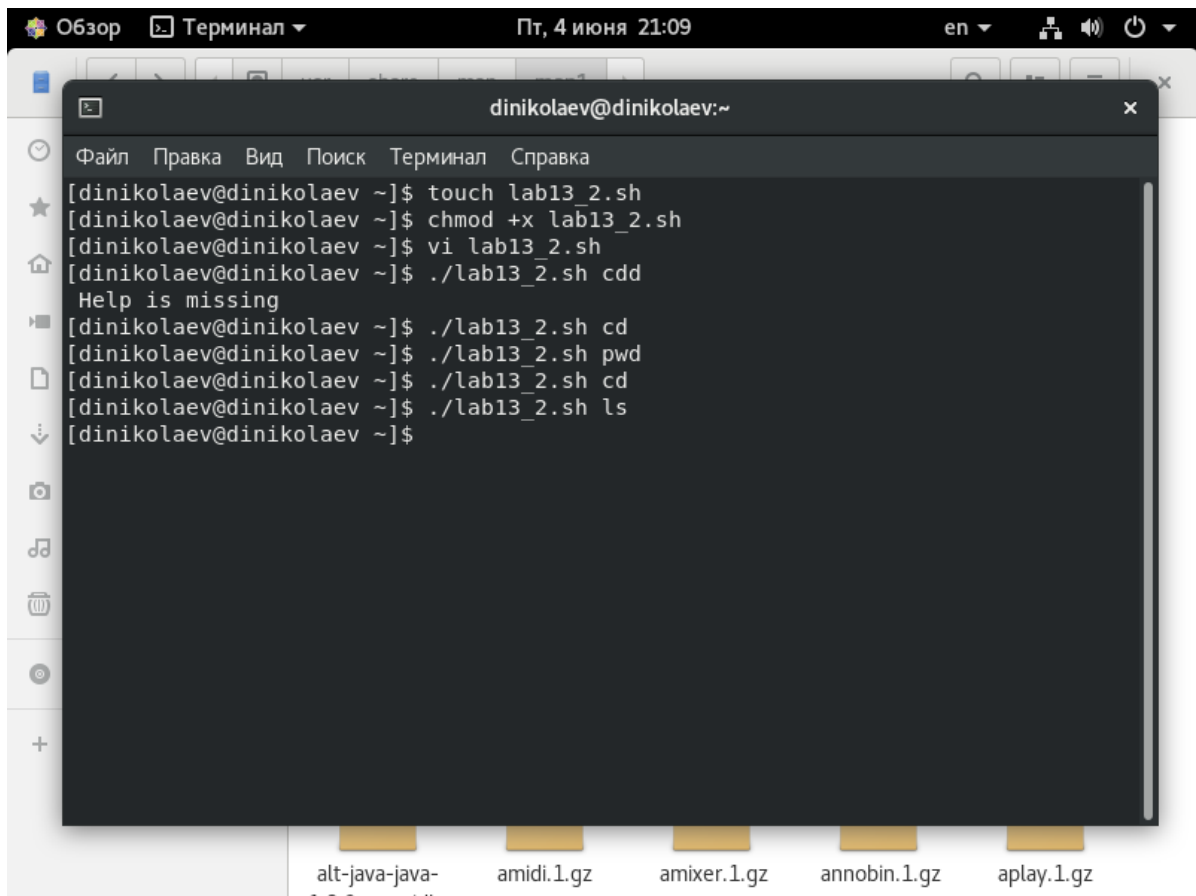
The screenshot shows a terminal window titled "dinikolaev@dinikolaev:~" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays a shell script for semaphore simulation. The script starts with a shebang line, sets a lockfile path, and uses a while loop with flock to manage access. It includes echo statements for "Lock", "Use of lockfile", and "Waiting", along with sleep commands. The script ends with a "done" statement. At the bottom of the terminal, there is a status bar with "-- ВСТАВКА --", a line number "14,2-9", and the word "Весь". Below the terminal window, a file manager shows several files: "alt-java-java-1.8.0-amd64", "amidi.1.gz", "amixer.1.gz", "annobin.1.gz", and "aplay.1.gz".

```
#!/bin/bash
lockfile="./lockfile"
exec {fn}>${lockfile}
echo " Lock"
while test -f lockfile
do
    if flock -n ${fn}
    then
        echo " Use of lockfile"
        sleep 3
    else
        echo " Waiting"
        sleep 1
    fi
done
```

- Командный файл, реализующий механизм семафоров







```
dinikolaev@dinikolaev:~  
Файл Правка Вид Поиск Терминал Справка  
[dinikolaev@dinikolaev ~]$ touch lab13_2.sh  
[dinikolaev@dinikolaev ~]$ chmod +x lab13_2.sh  
[dinikolaev@dinikolaev ~]$ vi lab13_2.sh  
[dinikolaev@dinikolaev ~]$ ./lab13_2.sh cdd  
Help is missing  
[dinikolaev@dinikolaev ~]$ ./lab13_2.sh cd  
[dinikolaev@dinikolaev ~]$ ./lab13_2.sh pwd  
[dinikolaev@dinikolaev ~]$ ./lab13_2.sh cd  
[dinikolaev@dinikolaev ~]$ ./lab13_2.sh ls  
[dinikolaev@dinikolaev ~]$
```

- Вывод на несуществующую команду и вызов справки по другим командам





The image shows a terminal window titled "dinikolaev@dinikolaev:~" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays the following script:

```
#!/bin/bash
echo "Random latinian word: "
for symbol in {A..Z} {a..z};
do SYMBOLS=$SYMBOLS$symbol;
done
String_Length=25
String=""
for i in `seq 1 $String_Length`
do
    String=$String${SYMBOLS:$(expr $RANDOM % ${#SYMBOLS}):1}
done
echo $String
```

Below the script, there are several tilde (~) characters representing the output of the script. At the bottom of the terminal, a status bar shows "lab13\_3.sh" 12L, 243C, "12,12", and "Весь". Below the terminal window, a file manager shows a list of files: "alt-java-java-1.8.0-openjdk", "amidi.1.gz", "amixer.1.gz", "annobin.1.gz", and "aplay.1.gz".

- Командный файл, выводящий случайную последовательность латинских букв

```
dinikolaev@dinikolaev:~  
[dinikolaev@dinikolaev ~]$ vi lab13_3.sh  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
EGGhIfFHGbCBctH0PTTtvTPmY  
[dinikolaev@dinikolaev ~]$ vi lab13_3.sh  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
PVBpXUdUIynNmfrbzuyzBWSlp  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
kcemTwvHVzKOGYibmncMtukVx  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
RSCnbaGpFbEMNQEkXuQGRptgA  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
QCgNvcBTzaTRXlCgerrPCSwW0  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
ycZPdLbXryEzEZMTQliiCzDVM  
[dinikolaev@dinikolaev ~]$ ./lab13_3.sh  
Random latinian word:  
PINPtzbFTHFajvEpyliRwfhn0  
[dinikolaev@dinikolaev ~]$
```

- Вывод командного файла

## 2.1 Контрольные вопросы

1. В этой строке надо квадратные скобки заменить на круглые ( `while ($1 != "exit")` ).
2. Можно объединить с помощью знаков `>`, `|` или используя подобные выражения `expr3=expr1expr2` (Использовалось в командном файле с randomными числами).
3. Команда `seq` выводит последовательность целых или действительных чисел с указанным шагом, можно использовать с циклом `for`, подставляя команды.

4. Результатом будет число 3 (дробная часть отбрасывается).
5. В `zsh` возможно свободно настраивать сочетания клавиш; встроенная команда `zmv` позволяет массово переименовывать файлы или директории; в `zsh` поддерживаются числа с плавающей точкой; в `zsh` нумерация начинается с 1.
6. Да, синтаксис этой конструкции верен (`for((a=1;a <= LIMIT;a++))`).
7. Перечислим некоторые отличия:
  - Скорость работы программ на ассемблере на 50% медленнее оптимизированных на C/C++;
  - Скорость работы виртуальной ява-машины с байт-кодом обычно превосходит скорость аппаратуры с кодами (уступает только ассемблеру и лучшим оптимизирующим трансляторам), получаемыми трансляторами с языков высокого уровня;
  - Скорость компиляции и исполнения программ на яваскрипт в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и даже превосходит некоторые качественные компиляторы (намного при этом обгоняя в скорости исполнения программ);
  - Скорость кодов, генерируемых компилятором C оказалась меньшей, чем у GNU;
  - Стек большинства тестируемых языков поддерживает только очень ограниченное число рекурсивных вызовов, некоторые же трансляторы позволяют увеличить размер стека;
  - В версиях `gawk`, `php`, `perl` и `bash` реализован динамический стек, который позволяет использовать всю ОП компьютера. Но `bash` использует стек слишком экстенсивно.

## 3 Выводы

Закрепил основы программирования в оболочке ОС UNIX/Linux, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.