

# **Лабораторная работа №14**

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

Буллер Татьяна Александровна

# Содержание

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Цель работы</b>                       | <b>4</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b>    | <b>5</b>  |
| 2.1      | Семафоры . . . . .                       | 5         |
| 2.2      | Map с помощью командного файла . . . . . | 8         |
| 2.3      | Случайная комбинация . . . . .           | 9         |
| <b>3</b> | <b>Выводы</b>                            | <b>11</b> |

## Список иллюстраций

|     |   |    |
|-----|---|----|
| 2.1 | Скрипт семафора . . . . .               | 6  |
| 2.2 | Перевод вывода в другое окно . . . . .  | 7  |
| 2.3 | Результат выполнения скрипта . . . . .  | 8  |
| 2.4 | Скрипт map . . . . .                    | 9  |
| 2.5 | Результат работы скрипта . . . . .      | 9  |
| 2.6 | Скрипт и результат выполнения . . . . . | 10 |

# 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**

### **2.1 Семафоры**

Необходимо написать командный файл, реализующий упрощённый механизм семафоров. В случае, если файл уже используется некоторым другим процессом, командный файл ожидает его освобождения 5 секунд и выдает соответствующее сообщение. Затем, если файл не освободился, цикл повторяется. Если же файл после ожидания стал свободен, то выводится сообщение о записи в файл и в сам файл записывается некоторая фраза.

```
1 #!/bin/bash
2 while test -f lockfile
3 do
4 sleep 5
5 echo "waiting"
6 done
7
8 touch lockfile
9 let c=10
10 while ((c-=1))
11 do
12 echo "writing"
13 echo "in file">>lockfile
14 sleep 7
15 done
16
17 rm lockfile
18
```

Рис. 2.1: Скрипт семафора

Проверим работу файла: откроем два окна терминала и в одном из них запустим файл в привелигированном режиме, а во втором - в фоновом, переводя вывод в первое окно. Для перевода вывода в окно графического терминала используем команду `> /dev/pts/number`, где `number` - номер графического интерфейса.

A terminal window with a dark background. The prompt is `(tabuller@jordi)-[~]`. The command `$ bash 1.sh > /dev/pts/1 &` is entered. The output `[1] 7629` is shown on the next line.

```
(tabuller@jordi)-[~]  
$ bash 1.sh > /dev/pts/1 &  
[1] 7629
```

Рис. 2.2: Перевод вывода в другое окно

Видим, что запущенный в первом окне файл первое время производит запись без проблем. Потом, когда подключается второй процесс, один из файлов начинает выводить сообщения об ожидании, когда файл записи оказывается занят.

```
(tabuller@jordi)-[~]  
$ bash 1.sh  
writing  
writing  
waiting  
waiting  
writing  
waiting  
writing  
waiting  
waiting  
writing  
^C  
  
(tabuller@jordi)-[~]  
$ waiting  
  
(tabuller@jordi)-[~]  
$ cat lockfile  
in file  
in file  
in file  
in file  
in file
```

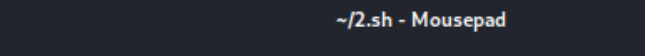
Рис. 2.3: Результат выполнения скрипта

## 2.2 Man с помощью командного файла

содержимое каталога /usr/share/man/man1 - архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд.



На некоторых дистрибутивах Linux каждый архив можно открыть командой `less`, сразу же просмотрев содержимое справки, однако в случае Kali эта опция `less` по умолчанию отключена. Для того, чтобы открыть архив `.gz` на Kali использую `zcat` и перевожу вывод в `less`. В качестве аргумента передаем название программы, которое вставится в код скрипта.



The screenshot shows a terminal window with the title bar '~/.sh - Mousepad'. The menu bar includes File, Edit, Search, View, Document, and Help. The toolbar contains icons for file operations (new, open, save, save as, print, close) and editing (undo, redo, cut, copy, paste, find, replace, refresh). The terminal content shows a prompt '1 #!/bin/bash' followed by a command '2 zcat /usr/share/man/man1/\$1.1.gz | less' and a prompt '3'.

Рис. 2.4: Скрипт man

```

tubuller@jordi: ~
File Actions Edit View Help

.\" Copyright (C) 1999-2011, 2013-2023 Free Software Foundation, Inc.
.\"
.\" This document is dual-licensed. You may distribute and/or modify it
.\" under the terms of either of the following licenses:
.\"
.\" * The GNU General Public License, as published by the Free Software
.\" Foundation, version 3 or (at your option) any later version. You
.\" should have received a copy of the GNU General Public License
.\" along with this program. If not, see
.\" <https://www.gnu.org/licenses/>.
.\"
.\" * The GNU Free Documentation License, as published by the Free
.\" Software Foundation, version 1.2 or (at your option) any later
.\" version, with no Invariant Sections, no Front-Cover Texts, and no
.\" Back-Cover Texts. You should have received a copy of the GNU Free
.\" Documentation License along with this program. If not, see
.\" <https://www.gnu.org/licenses/>.
.\"
.TH NANO 1 "version 7.2" "January 2023"

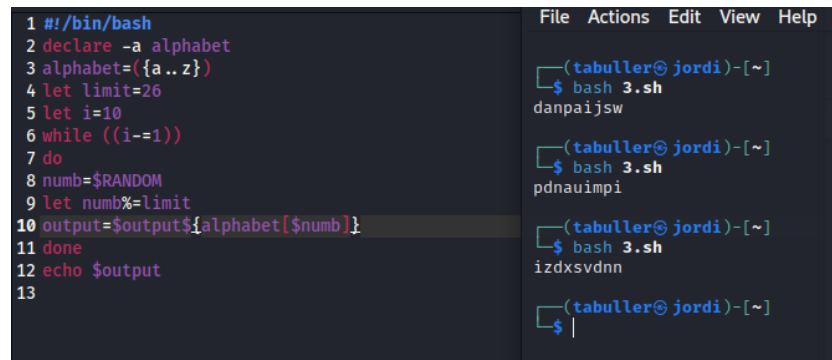
```

Рис. 2.5: Результат работы скрипта

## 2.3 Случайная комбинация

Зададим переменную `alphabet`: массив, который заполним латинскими маленькими буквами (`a..z`). Далее зададим переменную-ограничитель и проведем цикл по ней: на каждой итерации в переменную `numb` запишем случайное число, которое ограничим установленным до этого лимитом. Заполним получен-

ными элементами переменную-массив вывода и вызовом ее в конце программы:



```
1 #!/bin/bash
2 declare -a alphabet
3 alphabet={a..z}
4 let limit=26
5 let i=10
6 while ((i-=1))
7 do
8   numb=$RANDOM
9   let numb%=limit
10  output=$output${alphabet[numb]}
11 done
12 echo $output
13
```

The right side of the image shows four terminal sessions. Each session starts with a prompt '(tabuller@jordi)-[~]' followed by a '\$' prompt and the command 'bash 3.sh'. The first three sessions show the output of the script: 'danpaijsw', 'pdnauimpi', and 'izdxsvdnn'. The fourth session shows the prompt '\$ |' without any output.

Рис. 2.6: Скрипт и результат выполнения

## **3 Выводы**

Изучены основы программирования в оболочке ОС UNIX/Linux. Написаны более сложные командные файлы с использованием логических управляющих конструкций и циклов.