

# **Отчёт по лабораторной работе №11**

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

**Тагиев Павел Фаикович**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Задание 1 . . . . .	7
3.2	Задание 2 . . . . .	10
3.3	Задание 3 . . . . .	13
<b>4</b>	<b>Контрольные вопросы</b>	<b>15</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

## Список иллюстраций

3.1	Запуск скрипта в первом терминале . . . . .	8
3.2	Запуск скрипта во втором терминале . . . . .	9
3.3	Запуск трех и более процессов . . . . .	9
3.4	Изучение директории /usr/share/man/man1 . . . . .	11
3.5	Вывод напрямую через less . . . . .	11
3.6	Запуск реализации команды man . . . . .	12
3.7	Обработанный вывод мануала . . . . .	12
3.8	Пример использования . . . . .	13

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов [1].

## 2 Задание

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 имела возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.

Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

## 3 Выполнение лабораторной работы

### 3.1 Задание 1

---

Листинг 3.1 Упрощенный механизм семафоров

---

```
#!/usr/bin/bash

semaphore_wait() {
    # атомарная проверка того что ресурс не занят
    while ! mkdir /tmp/$0.lock 2> /dev/null; do
        sleep 0.5 # ждем освобождения ресурса
    done
}

semaphore_signal() {
    rmdir /tmp/$0.lock
}

echo -e "\nпроцесс номер $$ пытается захватить ресурс"

semaphore_wait # попытка войти в критическую секцию

echo -e "\nпроцесс номер $$ начал выполнять критическую секцию"

sleep 2 # выполнение какой-то работы требующей синхронизации

semaphore_signal # выход из критической секции

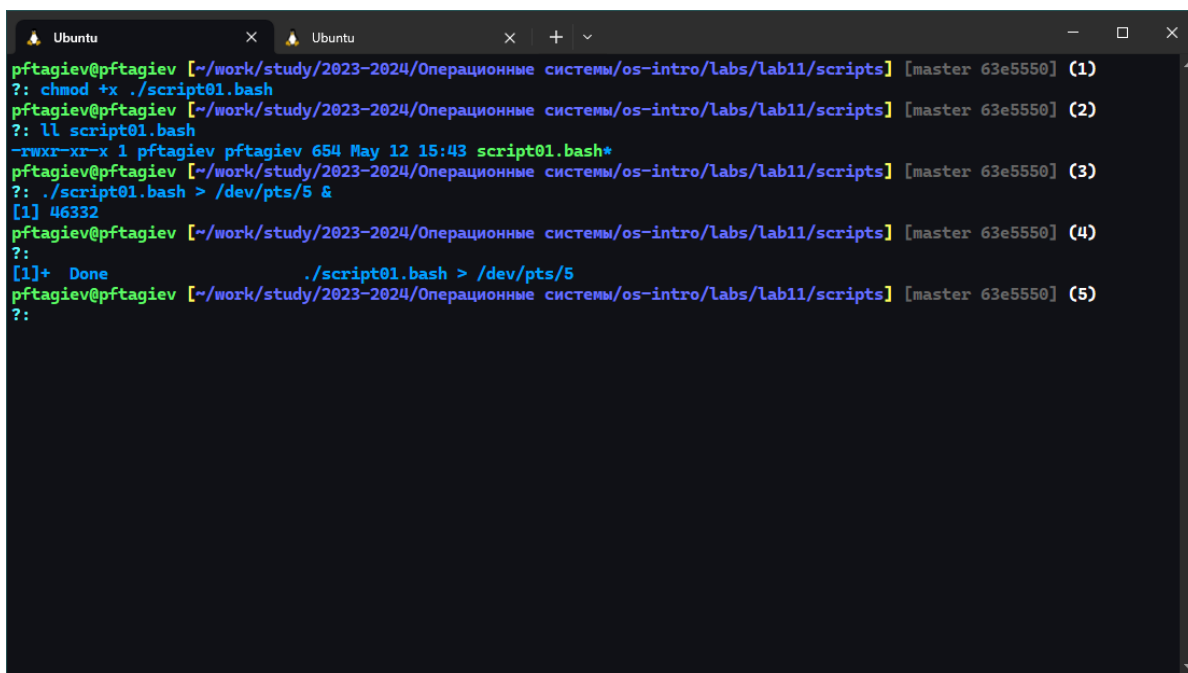
echo -e "\nпроцесс номер $$ освободил ресурс"
```

---

На лист. 3.1 можно увидеть простейшую реализацию бинарного семафора или

мьютекса [2]. В качестве атомарной проверки внутри функции `semaphore_wait` я использую команду `mkdir` [3]. Эта команда удобна тем что при попытке создания директории с существующим именем она возвращает ошибку.

Проверка существования директории, ее создание при отсутствии и возврат кода в рамках одной команды делает `mkdir` прекрасной альтернативой атомарной инструкции CAS [4] из других языков.



```
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (1)
?: chmod +x ./script01.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (2)
?: ll script01.bash
-rwxr-xr-x 1 pftagiev pftagiev 654 May 12 15:43 script01.bash*
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (3)
?: ./script01.bash > /dev/pts/5 &
[1] 46332
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (4)
?:
[1]+  Done                  ./script01.bash > /dev/pts/5
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (5)
?:
```

Рис. 3.1: Запуск скрипта в первом терминале

Начнем выполнение первой части задания. Для удобства сделаем скрипт исполняемым как показано на рис. 3.1 в промпте (1), затем запустим его в фоне перенаправив вывод в терминал `dev/pts/5`. В терминале `dev/pts/5` запустим тот же командный файл в обычном режиме, как показано на рис. 3.2.

Как видно все на том же рис. 3.2 семафор работает корректно и критическую секцию исполняет только один запущенный экземпляр скрипта, пока другой ждет.



```
Ubuntu
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (1)
?: tty
/dev/pts/5
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (2)
?: ./script01.bash
процесс номер 46332 пытается захватить ресурс
процесс номер 46332 начал выполнять критическую секцию
процесс номер 46344 пытается захватить ресурс
процесс номер 46332 освободил ресурс
процесс номер 46344 начал выполнять критическую секцию
процесс номер 46344 освободил ресурс
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (3)
?:
```

Рис. 3.2: Запуск скрипта во втором терминале

```
Ubuntu
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (1)
?: ./script01.bash & ./script01.bash & ./script01.bash & ./script01.bash & sleep 15
[1] 47606
[2] 47607
[3] 47608
[4] 47609
процесс номер 47606 пытается захватить ресурс
процесс номер 47609 пытается захватить ресурс
процесс номер 47606 начал выполнять критическую секцию
процесс номер 47608 пытается захватить ресурс
процесс номер 47607 пытается захватить ресурс
процесс номер 47606 освободил ресурс
процесс номер 47608 начал выполнять критическую секцию
процесс номер 47608 освободил ресурс
процесс номер 47607 начал выполнять критическую секцию
процесс номер 47607 освободил ресурс
процесс номер 47609 начал выполнять критическую секцию
процесс номер 47609 освободил ресурс
[1] Done ./script01.bash
[2] Done ./script01.bash
[3]- Done ./script01.bash
[4]+ Done ./script01.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (2)
?:
```

Рис. 3.3: Запуск трех и более процессов

Пойдем дальше и запустим сразу четыре экземпляра скрипта, как показано на рис. 3.3 в промте (1). Тут `sleep 15` нужен чтобы терминал не создавал новый промт 15 секунд и не мешал выводу запущенных на фоне процессов. Как видно семафор отлично справляется с тремя и более процессами.

## 3.2 Задание 2

---

### Листинг 3.2 Реализация команды `man`

---

```
#!/usr/bin/bash

file=/usr/share/man/man1/$1.1.gz # путь к мануалам

if [ -f $file ]; then
    # если архив существует
    # читаем его и передаем
    # интерпретатору языка разметки roff
    # затем читаем его постранично
    zcat $file | groff -man -T utf8 -i | less
else
    echo Для $1 не существует справки
fi
```

---

В этом задании требуется написать простую реализацию команды `man`, для этого изучим содержимое директории `/usr/share/man/man1`, в которой хранятся архивы мануалов команд. Как можно увидеть на рис. 3.4 имя архива имеет следующий шаблон: имя\_команды.1.gz.

Однако открывая архив просто передав его команде `less`, как предлагается в задании, я получил совсем не тот результат что ожидал. Как видно на рис. 3.5 файлы мануалов написаны с использованием специального языка разметки `roff`. Поэтому, чтобы добиться сопоставимого с командой `man` вывода, прежде чем печатать файлы мануалов на экран их нужно передать специальному интерпретатору `groff`, что я и сделал в своей реализации команды `man` на лист. 3.2.

```
Ubuntu
pftagiev@pftagiev [~] (1)
?: cd /usr/share/man/man1
pftagiev@pftagiev [/usr/share/man/man1] (2)
?: ls | head -10
CA.pl.1ssl.gz
GET.1p.gz
HEAD.1p.gz
JxrDecApp.1.gz
JxrEncApp.1.gz
NF.1.gz
POST.1p.gz
Xorg.1.gz
Xserver.1.gz
Xvfb.1.gz
pftagiev@pftagiev [/usr/share/man/man1] (3)
?: ls | grep "man"
man-recode.1.gz
man.1.gz
manconv.1.gz
manifest.1.gz
manpath.1.gz
pod2man.1.gz
pftagiev@pftagiev [/usr/share/man/man1] (4)
?: less man.1.gz
pftagiev@pftagiev [/usr/share/man/man1] (5)
?:
```

Рис. 3.4: Изучение директории /usr/share/man/man1

```
Ubuntu
.\" t
.\" ** The above line should force tbl to be a preprocessor **
.\" Man page for man
.\"
.\" Copyright (C) 1994, 1995, Graeme W. Wilford. (Wilf.)
.\" Copyright (C) 2001-2019 Colin Watson.
.\"
.\" You may distribute under the terms of the GNU General Public
.\" License as specified in the file COPYING that comes with the
.\" man-db distribution.
.\"
.\" Sat Oct 29 13:09:31 GMT 1994  Wilf. (G.Wilford@ee.surrey.ac.uk)
.\"
.pc
.TH MAN 1 "2022-03-17" "2.10.2" "Manual pager utils"
.SH NAME
man \- an interface to the system reference manuals
.SH SYNOPSIS
.\" The general command line
.B man
.RI [\| "man options" \|]
.RI [\|[\| section \|]
.IR page \ \|.\|.\|. \|] \ \|.\|.\|. \|&
.\" The apropos command line
.br
.B man
.B \-k
.RI [\| "apropos options" \|]
.I regexp
man.1.gz
```

Рис. 3.5: Вывод напрямую через less

Запустив данную реализацию команды `man`, как показано на рис. 3.6, мы получим красивую, обработанную справку к реальной команде `man` (рис. 3.7).

```
Ubuntu
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (1)
?: bash script02.bash man
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (2)
?:
```

Рис. 3.6: Запуск реализации команды man

```
MAN(1) Manual pager utils MAN(1)

NAME
  man - an interface to the system reference manuals

SYNOPSIS
  man [man options] [[section] page ...] ...
  man -k [apropos options] regexp ...
  man -K [man options] [section] term ...
  man -f [whatis options] page ...
  man -l [man options] file ...
  man -w|-W [man options] page ...

DESCRIPTION
  man is the system's manual pager. Each page argument given to man is
  normally the name of a program, utility or function. The manual page
  associated with each of these arguments is then found and displayed. A
  section, if provided, will direct man to look only in that section of
  the manual. The default action is to search in all of the available
  sections following a pre-defined order (see DEFAULTS), and to show only
  the first page found, even if page exists in several sections.

  The table below shows the section numbers of the manual followed by the
  types of pages they contain.

  tab (@); 1 lx. 1@T{ Executable programs or shell commands T} 2@T{ Sys-
```

Рис. 3.7: Обработанный вывод мануала

## 3.3 Задание 3

---

### Листинг 3.3 Генерация случайной последовательности

---

```
#!/usr/bin/bash

lowercase=(a b c d e f g h i j k l m n o p q r s t u v w x y z)
uppercase=(A B C D E F G H I J K L M N O P Q R S T U V W X Y Z)

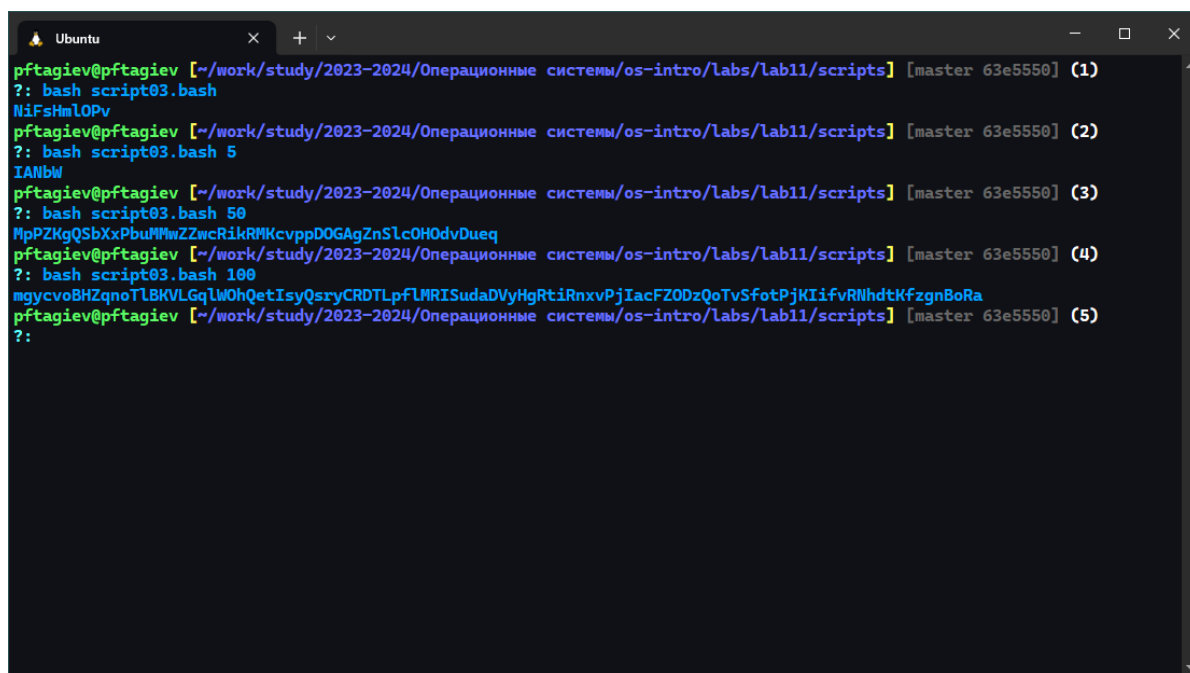
# все буквы английского алфавита
letters=("${lowercase[@]}" "${uppercase[@]}")

result=

# по умолчанию длина последовательности 10
for ((i=${1:-10}; i-->0)); do
    result+="${letters[$RANDOM%${#letters[@]}]}"
done

echo $result
```

---



The screenshot shows a terminal window with the following content:

```
Ubuntu
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (1)
?: bash script03.bash
NiFsHmLOPv
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (2)
?: bash script03.bash 5
IANbw
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (3)
?: bash script03.bash 50
MpPZKgQsBxPbuMMwZZwcRikRMKcvppDOGAgnZnSlcOH0dvDueq
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (4)
?: bash script03.bash 100
mgycvoBHZqnoTLBKVLGqLWOhQetIsyQsryCRDTLPfLMRISudaDVyHgRtiRnxvPjIacFZODzQoTvSfotPjKIifvRNhdtKfzgnBoRa
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab11/scripts] [master 63e5550] (5)
?:
```

Рис. 3.8: Пример использования

Следуя заданию, напишем командный файл, который генерирует случайную последовательность букв английского алфавита используя встроенную переменную `$RANDOM`. Реализацию можно увидеть на лист. 3.3, а пример использования на рис. 3.8.

## 4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

```
while [$1 != "exit"]
```

Ошибка в том что после [ и до ] должен идти пробел, еще имеет смысл обернуть `$1` в *двойные кавычки*, чтобы эта переменная корректно обрабатывалась даже если она содержит пробелы.

2. Как объединить (конкатенировать) несколько строк в одну?

Есть множество способов конкатенации строк в *bash*, я разберу два основных. Предположим у нас есть переменные `a` и `b`, чтобы добавить к строке `a` строку `b` справа можно использовать следующий синтаксис: `a+=$b`. Записать результат конкатенации в новую переменную можно так: `c="$a$b"`.

3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на *bash*?

Утилита `seq` используется для генерации последовательностей, ее можно использовать в циклах. Например: `for i in $(seq 0 2 10)`, переменная `i` будет принимать значения от 0 до 10 включительно с шагом 2. Очевидно, что чтобы добиться похожего результата можно использовать *Cи-подобный* цикл `for`: `for ((i=0;i<=10;i+=2))`

4. Какой результат даст вычисление выражения `$((10/3))`?

Результат будет 3, так как будет выполнено целочисленное деление.

5. Укажите кратко основные отличия командной оболочки *zsh* от *bash*.

- *Интерактивность*: *zsh* известен своими улучшенными интерактивными

возможностями, такими как более продвинутый автодополнение команд и файлов, что делает его более удобным для интерактивного использования.

- *Темы и плагины*: *zsh* поддерживает темы и плагины, что позволяет пользователям настраивать свой рабочий интерфейс и расширять функциональность оболочки.
- *Синтаксис*: *zsh* обладает более гибким синтаксисом, включая улучшенные возможности для работы с массивами и ассоциативными массивами.
- *Совместимость*: *zsh* во многом совместим с *bash*, но включает множество дополнительных возможностей, которые могут не работать в *bash* без изменений.
- *Контекстное автодополнение*: *zsh* предлагает более продвинутое контекстное автодополнение, которое может учитывать не только имена файлов и команд, но и их параметры.
- *Модульность*: *zsh* разработан с учётом модульности, что позволяет легко добавлять новые функции и интегрировать внешние скрипты.

6. Проверьте, верен ли синтаксис данной конструкции

```
for ((a=1; a <= LIMIT; a++))
```

Да, синтаксис верен. В двойных круглых скобках можно использовать переменные без знака доллара.

7. Сравните язык *bash* с какими-либо языками программирования. Какие преимущества у *bash* по сравнению с ними? Какие недостатки?

- Преимущества *bash*:
  - *Скорость разработки*: *bash*-скрипты обычно проще и быстрее писать для простых задач, особенно для автоматизации командной строки и системных операций.
  - *Встроенная поддержка UNIX-команд*: *bash* нативно интегрируется с *UNIX*-командами и утилитами, что делает его мощным



инструментом для системного администрирования.

- *Портативность*: *bash*-скрипты легко переносить между различными *UNIX*-подобными системами без изменений.
- Недостатки *bash*:
  - *Ограниченные возможности программирования*: *bash* не имеет такого богатого набора функций программирования, как *C++* или *Python*, например, объектно-ориентированное программирование или обширные стандартные библиотеки.
  - *Медленная производительность*: Для сложных задач или задач, требующих интенсивных вычислений, *bash* может работать медленнее, чем *C++* или *Python*.
  - *Сложность синтаксиса*: Некоторые аспекты синтаксиса *bash* могут быть непривычными или запутанными для новичков, особенно при работе с текстовыми строками и файлами.

## 5 Выводы

В этой лабораторной работе мы закрепили свои знания об основах программирования на языке *bash*. Написав простейшие реализации семафора и команды `man`, используя переменную `$RANDOM` для генерации последовательности букв английского алфавита.

## Список литературы

1. Кулябов. Операционные системы. Москва: РУДН, 2016. 118 с.
2. Semaphore (programming) [Электронный ресурс]. 2024. URL: [https://en.wikipedia.org/wiki/Semaphore\\_\(programming\)](https://en.wikipedia.org/wiki/Semaphore_(programming)).
3. How can I ensure that only one instance of a script is running at a time (mutual exclusion, locking)? [Электронный ресурс]. 2024. URL: <https://mywiki.woledge.org/BashFAQ/045>.
4. Compare-and-swap [Электронный ресурс]. 2024. URL: <https://en.wikipedia.org/wiki/Compare-and-swap>.