

# **Отчёт по лабораторной работе №13**

**Операционные системы**

Ильина Любовь Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>13</b>

## **Список таблиц**

## Список иллюстраций

2.1	Командный файл . . . . .	7
2.2	Запуск вывода файла в текстовую консоль . . . . .	7
2.3	Исполнение командного файла в текстовой консоли . . . . .	8
2.4	Командный файл . . . . .	9
2.5	Исполнение командного файла . . . . .	9
2.6	Командный файл . . . . .	10
2.7	Исполнение командного файла . . . . .	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

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. (рис. 2.1 - 2.3)

```
lailjina@lailjina:~/work/study/2022-2023/Операционные системы/os-intro
File Edit View Search Terminal Help
#!/bin/bash
semaphore_file="semaphore.txt"
wait_time=5

usage_time=10

if [ ! -f "$semaphore_file" ]; then
    echo 0 > "$semaphore_file"
fi

wait_for_resource() {
    while true; do
        semaphore_value=$(cat "$semaphore_file")

        if [ "$semaphore_value" -eq 0 ]; then
            break
        fi

        echo "Ресурс занят. Ожидание освобождения..."
        sleep $wait_time
    done
    echo 1 > "$semaphore_file"
}

use_resource() {
    echo "Ресурс используется процессом $$"

    sleep $usage_time

    echo 0 > "$semaphore_file"

    echo "Ресурс освобожден"
}

wait_for_resource
use_resource
```

Рис. 2.1: Командный файл

```
[root@lailjina os-intro]# ./131.sh > /dev/tty3
```

Рис. 2.2: Запуск вывода файла в текстовую консоль

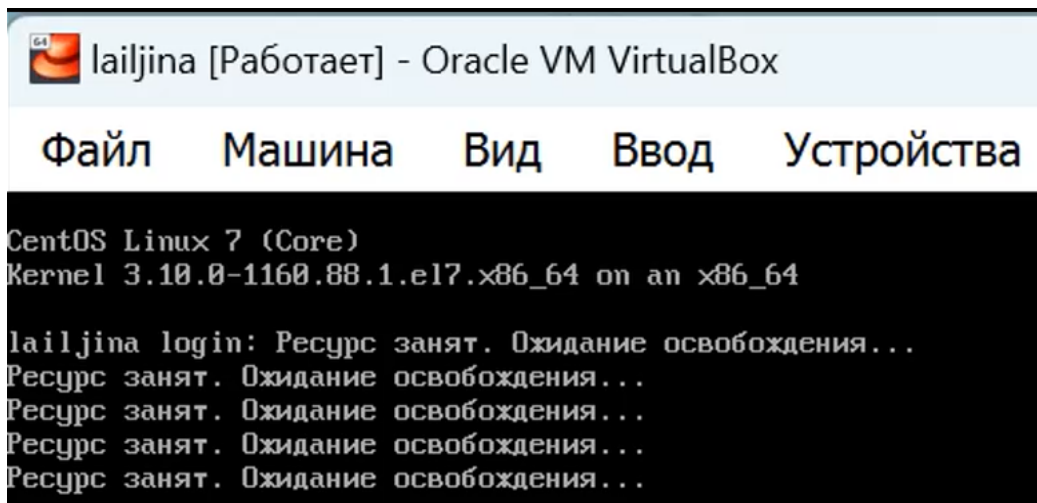


Рис. 2.3: Исполнение командного файла в текстовой консоли

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`. (рис. 2.4 - 2.5)



```

132.sh - emacs@lailjina.localdomain
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
command_name="$1"
man_directory="/usr/share/man/man1"

if [ -z "command_name" ]; then
    echo "Введите название команды в качестве аргумента."
    exit 1
fi

man_file="$man_directory/$command_name.1.gz"
if [ -f "man_file" ]; then
    less $man_file
else
    echo "Справка по команде '$command_name' не найдена."
fi

```

Рис. 2.4: Командный файл

```

lailjina@lailjina:/home/lailjina/work/study/2022-2023/Операционные системы/os-intro
File Edit View Search Terminal Help
LS(1) User Commands LS(1)

ESC[1mNAMEESC[0m
ls - list directory contents

ESC[1mSYNOPSISESC[0m
ESC[1mls ESC[22m[ESC[4mOPTIONESC[24m]... [ESC[4mFILEESC[24m]...

ESC[1mDESCRIPTIONESC[0m
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of ESC[1m-cftuvSUX ESC[22mnor ESC[1m-
-sort ESC[22mis speci-
fied.

Mandatory arguments to long options are mandatory for short options
too.

ESC[1m-aESC[22m, ESC[1m--allESC[0m
do not ignore entries starting with .

ESC[1m-AESC[22m, ESC[1m--almost-allESC[0m
do not list implied . and ..

ESC[1m--authorESC[0m
with ESC[1m-lESC[22m, print the author of each file

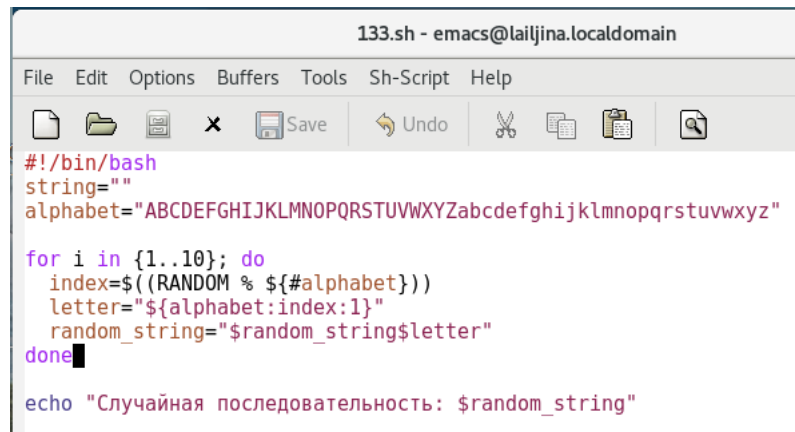
ESC[1m-bESC[22m, ESC[1m--escapeESC[0m
print C-style escapes for nongraphic characters

ESC[1m--block-sizeESC[22m, ESC[1m--block-size=BLOCK_SIZEESC[0m

```

Рис. 2.5: Исполнение командного файла

- 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 (рис. 2.6 - 2.7)



```
#!/bin/bash
string=""
alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

for i in {1..10}; do
    index=$((RANDOM % ${#alphabet}))
    letter="${alphabet:index:1}"
    random_string="$random_string$letter"
done

echo "Случайная последовательность: $random_string"
```

Рис. 2.6: Командный файл

```
[lailjina@lailjina os-intro]$ ./133.sh
Случайная последовательность: XQAeWkDpWD
```

Рис. 2.7: Исполнение командного файла

- Исправление синтаксической ошибки в строке while [\$1 != "exit"] : while [[ \$1 != "exit" ]]
- Как объединить (конкатенация) несколько строк в одну? с помощью + и = можно объединить несколько строк в одну: concatenated\_string="string1string2"
- Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash? Утилита seq в bash используется для генерации последовательностей чисел. Реализовать функционал возможно циклами:

```
for ((i=1; i<=10; i++))
current=$start
while [ $current -le $end ]
do
```

```
echo "$current"
current=$((current + 1))
done
```

4. Какой результат даст вычисление выражения  $\$(10/3)$ ? целочисленное деление даст результат 3.
5. Укажите кратко основные отличия командной оболочки zsh от bash. Конфигурационные файлы: Bash использует файлы `~/.bashrc` и `~/.bash_profile` для настройки окружения, а Zsh использует файлы `~/.zshrc` и `~/.zprofile`. У Zsh также есть более сложная и гибкая система настройки, позволяющая использовать файлы конфигурации, такие как `~/.zshrc`, `~/.zshenv`, `~/.zlogin` и другие.

Автодополнение: Zsh обладает мощным и гибким механизмом автодополнения, который предлагает подсказки и автодополнение команд, параметров и файлов. Bash также имеет функциональность автодополнения, но не такую мощную и настраиваемую, как в Zsh.

Синтаксис: Zsh имеет некоторые отличия в синтаксисе и некоторые дополнительные возможности по сравнению с Bash. Например, в Zsh поддерживается прямая подстановка переменных без использования кавычек, и существуют другие изменения в работе с параметрами командной строки.

6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))` верно
7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? Bash является командной оболочкой и языком сценариев (скриптовым языком), предназначенным для автоматизации задач в операционной системе Unix и Unix-подобных системах. Вот сравнение `bash` с Python:  
Преимущества `bash`:

- 1) Легко интегрируется с системными командами и утилитами.
- 2) Отлично подходит для выполнения системных задач и автоматизации.
- 3) Простой и быстрый для написания скриптов. Недостатки `bash`:

- 4) Ограниченные возможности для сложных алгоритмов и структур данных.
- 5) Отсутствие некоторых функций и библиотек, которые могут быть доступны в других языках, таких как Python.

### **3. Выводы**

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