

# **Отчёт по лабораторной работе №12**

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

Мокочунина Влада Сергеевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	15

## Список иллюстраций

3.1	Написание . . . . .	8
3.2	Проверка . . . . .	9
3.3	Написание . . . . .	10
3.4	Проверка . . . . .	11
3.5	Команда . . . . .	12
3.6	Написание . . . . .	13
3.7	Проверка . . . . .	13
3.8	Написание . . . . .	14
3.9	Проверка . . . . .	14

## 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

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

## **3 Выполнение лабораторной работы**

1. Написание файла. (рис. [3.1]).

```
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t1))
do
    echo "Waiting"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t < t2))
do
    echo "Execution"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
```

Рис. 3.1: Написание



## 2. Проверка работы

```
[vsmokochunina@10 ~]$ ./prog1.sh 4 7  
Waiting  
Waiting  
Waiting  
Waiting  
Execution  
Execution  
Execution  
Execution  
Execution  
Execution  
Execution  
[vsmokochunina@10 ~]$ emacs
```

Рис. 3.2: Проверка

## 3. Доработка файла

```
#!/bin/bash
function ogidanie
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t < t1))
    do

        echo "Waiting"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
function vipolnenie
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t < t2))
    do

        echo "Execution"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Exit" ]
    then
        echo "Exit"
        exit 0
    fi
    if [ "$command" == "Waiting" ]
    then ogidanie
    fi
    if [ "$command" == "Execution" ]
    then vipolnenie
    fi
    echo "Next action: "
    read command
done
```

~\$ cat prog1.sh

Рис. 3.3: Написание

## 4. Проверка работы

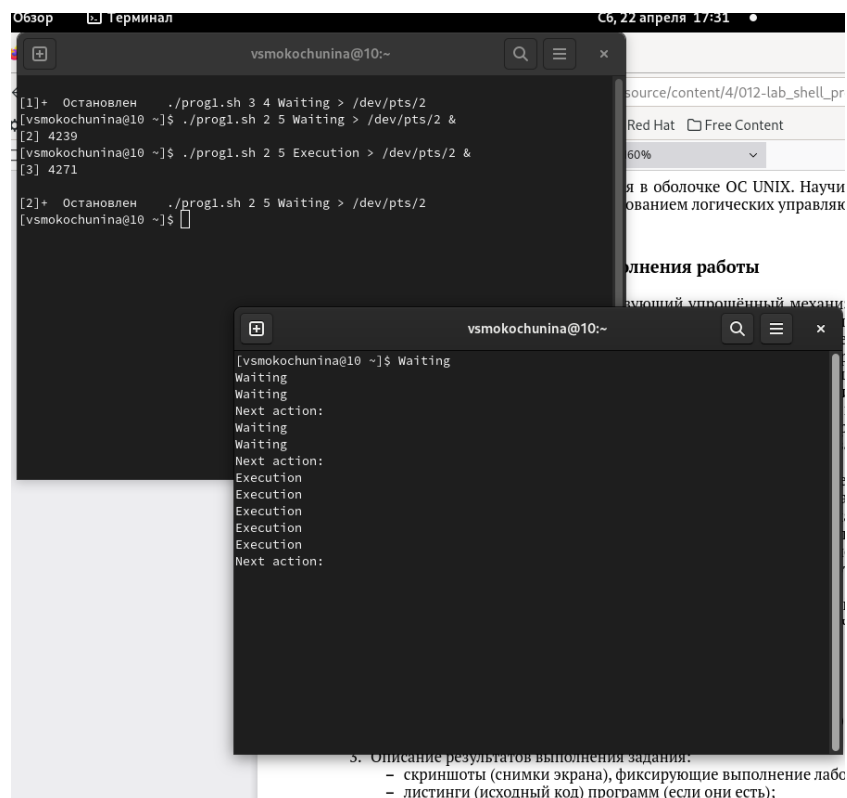


Рис. 3.4: Проверка

## 5. Команда

```
vsmokochunina@10:usr/share/n
[vsmokochunina@10 ~]$ cd /usr/share/man/man1
[vsmokochunina@10 man1]$ ls
.:1.gz
'.1.gz'
ab.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-core.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-oops-for-hw-error.1.gz
abrt-action-find-bodhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
abrt-action-install-debuginfo.1.gz
abrt-action-list-dsos.1.gz
abrt-action-notify.1.gz
abrt-action-perform-ccpp-analysis.1.gz
abrt-action-save-package-data.1.gz
abrt-action-trim-files.1.gz
abrt-applet.1.gz
abrt-auto-reporting.1.gz
abrt-bodhi.1.gz
abrt-cli.1.gz
abrt-dump-journal-core.1.gz
abrt-dump-journal-oops.1.gz
abrt-dump-journal-xorg.1.gz
abrt-dump-oops.1.gz
abrt-dump-xorg.1.gz
abrt-handle-upload.1.gz
abrt-harvest-pstoreoops.1.gz
abrt-harvest-vmcore.1.gz
abrt-merge-pstoreoops.1.gz
```

Рис. 3.5: Команда

## 6. Написание файла

```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
    gunzip -c /usr/share/man/man1/$1.1.gz | less
else
    echo "Справки по данной команде нет"
fi
```

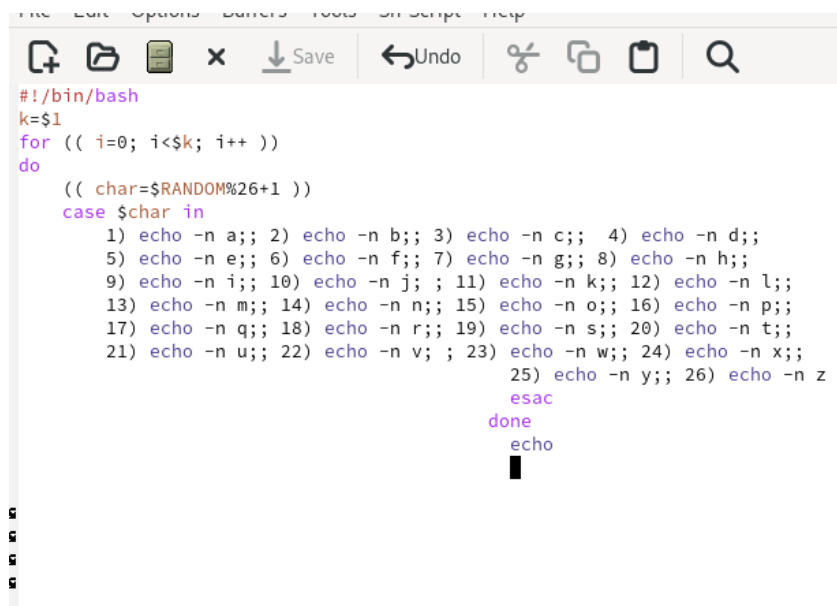
Рис. 3.6: Написание

## 7. Проверка работы

```
vsmokochunina@10:~ — /bin/bash ./man.sh ls
.\ DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "January 2023" "GNU coreutils 9.1" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[\fi,\OPTION\fR]... [\fi,\FILE\fR]...
.SH DESCRIPTION
.\ Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB-cftuvSUX\fR nor \fB-\fR is specified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB-a\fR, \fB-\fR
do not ignore entries starting with .
.TP
\fB-A\fR, \fB-\fR
do not list implied . and ..
.TP
\fB-\fR
with \fB-\fR, print the author of each file
.TP
\fB-b\fR, \fB-\fR
print C-style escapes for nongraphic characters
.TP
\fB-\fR
with \fB-\fR, scale sizes by SIZE when printing them;
e.g., '\fB-size=M'; see SIZE format below
.TP
\fB-B\fR, \fB-\fR
do not list implied entries ending with ~
.TP
\fB-c\fR
with \fB-\fR: sort by, and show, ctime (time of last
modification of file status information);
with \fB-\fR: show ctime and sort by name;
.
```

Рис. 3.7: Проверка

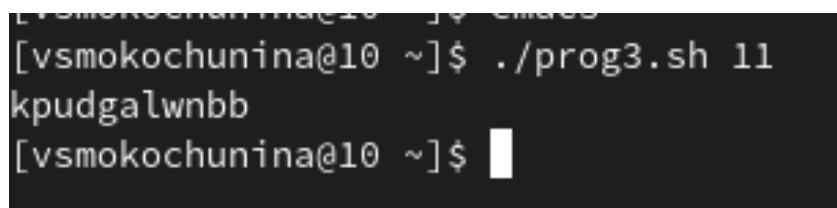
## 8. Написание файла

A screenshot of a text editor window with a menu bar (File, Edit, Options, Editors, Tools, Run Script, Help) and a toolbar with icons for file operations and editing. The editor contains a shell script with the following code:

```
#!/bin/bash
k=$1
for (( i=0; i<$k; 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. 3.8: Написание

## 9. Проверка работы

A screenshot of a terminal window showing the execution of a script. The prompt is [vsmokochunina@10 ~]\$. The command ./prog3.sh 11 is entered, and the output is kpudgalwnbb. The prompt is then [vsmokochunina@10 ~]\$ followed by a cursor.

```
[vsmokochunina@10 ~]$ ./prog3.sh 11
kpudgalwnbb
[vsmokochunina@10 ~]$
```

Рис. 3.9: Проверка

## 4 Выводы

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