

# Отчёт по лабораторной работе №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

---

Мокочунина В.С.

22 апреля 2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Мокочунина Влада Сергеевна
- Российский университет дружбы народов
- vmokochunina@gmail.com
- [https://github.com/Vmokochunina/study\\_2022-2023\\_os-intro.git](https://github.com/Vmokochunina/study_2022-2023_os-intro.git)

## Вводная часть

---

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

## Создание презентации

---

```
#!/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
```

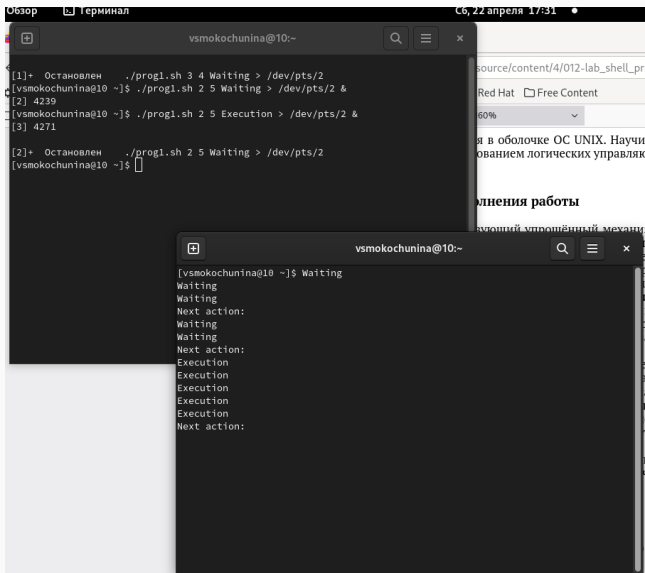
```
[vsmokochunina@10 ~]$ ./prog1.sh 4 7  
Waiting  
Waiting  
Waiting  
Waiting  
Execution  
Execution  
Execution  
Execution  
Execution  
Execution  
Execution  
[vsmokochunina@10 ~]$ emacs
```



```
#!/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
```



The image shows two terminal windows from a desktop environment. The top window, titled 'Обзор Терминал' and 'C6, 22 апреля 17:31', shows the execution of a script named 'prog1.sh' with three arguments: 3, 4, and 2. The output indicates the script is stopped ('Остановлен') and shows the process ID 4239. The bottom window, titled 'vsmokochunina@10:~', shows the execution of the same script with arguments 2, 5, and 2. The output shows the script is stopped ('Остановлен') and shows the process ID 4271. The background of the desktop shows a document titled 'source/content/4/012-lab\_shell\_pr' with text in Russian, including 'Red Hat Free Content', '60%', and 'я в оболочке ОС UNIX. Научи'.

```
vsmokochunina@10:~  
[1]+ Остановлен ./prog1.sh 3 4 Waiting > /dev/pts/2  
[vsmokochunina@10 ~]$ ./prog1.sh 2 5 Waiting > /dev/pts/2 &  
[2] 4239  
[vsmokochunina@10 ~]$ ./prog1.sh 2 5 Execution > /dev/pts/2 &  
[3] 4271  
  
[2]+ Остановлен ./prog1.sh 2 5 Waiting > /dev/pts/2  
[vsmokochunina@10 ~]$
```

```
vsmokochunina@10:~  
[vsmokochunina@10 ~]$ Waiting  
Waiting  
Waiting  
Next action:  
Waiting  
Waiting  
Next action:  
Execution  
Execution  
Execution  
Execution  
Execution  
Next action:
```

## 3. Описание результатов выполнения задания:

- скриншоты (снимки экрана), фиксирующие выполнение лаб
- вставки (исходный код) программ (если они есть):

```
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
```

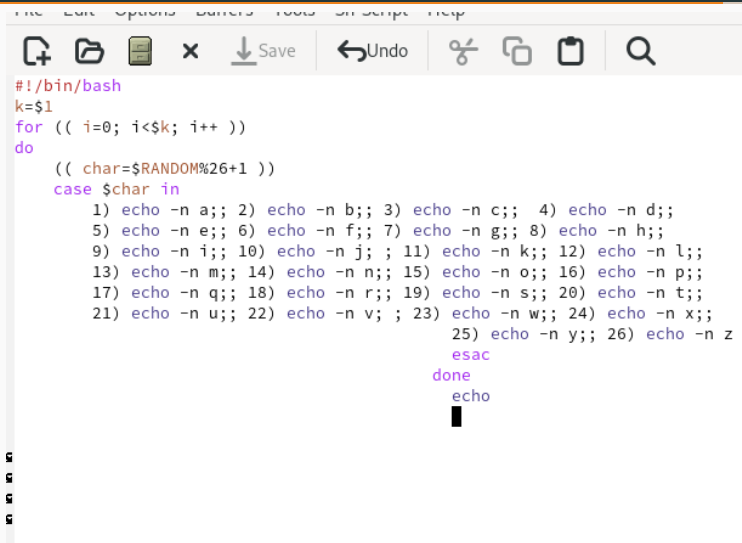
```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
    gunzip -c /usr/share/man/man1/$1.1.gz | less
else
    echo "Справки по данной команде нет"
fi
```

Рис. 6: Написание

# Проверка работы

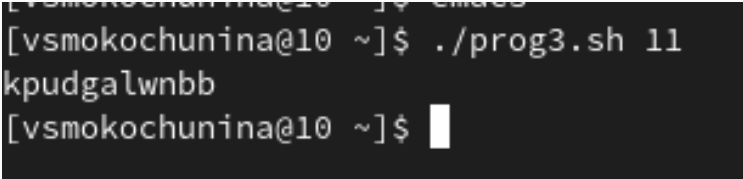
```
vsmokochunina@10:~ — /bin/bash ./man.sh ls

.\" 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\-\sort\fR is specified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-\a\fR, \fB\-\all\fR
do not ignore entries starting with .
.TP
\fB\-\A\fR, \fB\-\almost\-\all\fR
do not list implied . and ..
.TP
\fB\-\author\fR
with \fB\-\l\fR, print the author of each file
.TP
\fB\-\b\fR, \fB\-\escape\fR
print C\-\style escapes for nongraphic characters
.TP
\fB\-\block\-\size\fR=\fI\,SIZE\|\fR
with \fB\-\l\fR, scale sizes by SIZE when printing them;
e.g., '\fB\-\block\-\size=M\fR'; see SIZE format below
.TP
\fB\-\B\fR, \fB\-\ignore\-\backups\fR
do not list implied entries ending with ~
.TP
\fB\-\c\fR
with \fB\-\lt\fR: sort by, and show, ctime (time of last
modification of file status information);
with \fB\-\l\fR: show ctime and sort by name;
```



The screenshot shows a text editor window with a menu bar (File, Edit, Options, Editors, Tools, Shell Scripts, Help) and a toolbar with icons for file operations and editing. The main text area contains a shell script. The script starts with a shebang line, assigns a value to 'k', and enters a 'for' loop. Inside the loop, it generates a random character and uses a 'case' statement to echo characters from 'a' to 'z'. The script ends with 'done' and 'echo'.

```
#!/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
done
```



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

A terminal window with a dark background and light gray text. The prompt is [vsmokochunina@10 ~]\$. The command ./prog3.sh 11 is entered and executed, resulting in the output kpudgalwnbb. The prompt returns to [vsmokochunina@10 ~]\$.

Рис. 9: Проверка

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



