

# Программирование в командном процессоре ОС UNIX. Расширенное программирование

Лабораторная работа №12

---

Нати Ф. Б.

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Нати Франсиску Бунда
- студент 1 курса, группа НММбд-02-22
- Российский университет дружбы народов



## Вводная часть

---

- Командный процессор ОС UNIX
- Командные файлы

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

- Ознакомиться с теоретическим материалом.
- Выполнить упражнения.
- Ответить на контрольные вопросы.

## Выполнение лабораторной работы №12

---




# Первая программа

```
Открыть ▾ + lab12_1.sh
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%R)
s2=$(date +%R)
((t=$s2-$s1))
while ((t < t1)) do
    echo "Ожидайте"
    sleep 1
    s2=$(date +%R)
    ((t=$s2-$s1))
done
s1=$(date +%R)
s2=$(date +%R)
((t=$s2-$s1))
while ((t < t2)) do
    echo "Выполнение"
    sleep 1
    s2=$(date +%R)
    ((t=$s2-$s1))
done
t1=$1
t2=$2
```

```
[fbnati@fedora ~]$ chmod +x *.sh
```

```
[fbnati@fedora ~]$ ./lab12_1.sh 4 5
Ожидайте
Ожидайте
Ожидайте
Ожидайте
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
```

# Первая программа доработка

Открыть ▾  lab12\_1.sh  
~/

```
sleep 1
s2=$(date +%s)
((t=s2-$s1))
done
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then echo "Выход"
        exit 0
    fi
    if [ "$command" == "Ожидание" ]
    then pass
    fi
    if [ "$command" == "Выполнение" ]
    then pass
    fi
    echo "Следующее действие"
    read command
done
```

```
[fbnati@fedora ~]$ ./lab12_1.sh 4 5 3
```

```
Ожидайте
Ожидайте
Ожидайте
Ожидайте
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие

Следующее действие

Следующее действие
```

## Вторая программа

```
[fbnati@fedora ~]$ cd /usr/share/man/man1
[fbnati@fedora man1]$ ls
i.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
```

```
Открыть ▾  • lab12_2.sh
~
#!/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
```




## Вторая программа

```
[fbnati@fedora ~]$ ./lab12_2.sh ls
[fbnati@fedora ~]$ ./lab12_2.sh pwd
```

```
fbnati@fedora:~ — /bin/bash ./lab12_2.sh ls
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "March 2022" "GNU coreutils 9.0" "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
.\"
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB\--cftuvSUX\|fR nor \fB\--sort\|fR is sp
ecified.
.\"
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
```

```
fbnati@fedora:~ — /bin/bash ./lab12_2.sh pwd
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH PWD "1" "March 2022" "GNU coreutils 9.0" "User Commands"
.SH NAME
pwd \- print name of current/working directory
.SH SYNOPSIS
.B pwd
[fi,OPTION\|fR]...
.SH DESCRIPTION
.\" Add any additional description here
.\"
Print the full filename of the current working directory.
.TP
\fB\--L\|fR, \fB\--logical\|fR
use PWD from environment, even if it contains symlinks
.TP
\fB\--P\|fR, \fB\--physical\|fR
avoid all symlinks
.TP
\fB\--help\|fR
display this help and exit
.TP
\fB\--version\|fR
output version information and exit
```

# Третья программа

```
Открыть ▾  lab12_3.sh    
#!/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;;  
        ...  
    done  
done
```

```
[fbnati@fedora ~]$ chmod +x *.sh
```

```
[fbnati@fedora ~]$ ./lab12_3.sh 23  
mqhejanbfreixjuthwaas  
[fbnati@fedora ~]$ ./lab12_3.sh 3  
ggz
```

1. while [ \$1 != "exit" ] В данной строчке допущены следующие ошибки: • не хватает пробелов после первой скобки [и перед второй скобкой ] • выражение \$1 необходимо взять в " ", потому что эта переменная может содержать пробелы. Таким образом, правильный вариант должен выглядеть так: while [ "\$1" != "exit" ]

2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:

Первый: VAR1="Hello, "VAR2=" World" VAR3="❏ ❏❏1VAR2" echo "VAR3" :  
*Hello, World* : VAR1 = "Hello, "VAR1+ = "World"echo"VAR1" Результат: Hello, World

3. Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: seq LAST: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение is не выдает. seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT . Если LAST меньше, чем FIRST, он не производит вывод. seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными. seq -s «STRING» ПЕРВЫЙ

## Результаты

---

В ходе выполнения лабораторной работы были изучены основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.