

# Лабораторная работа № 14. Программирование в командном процессоре ОС UNIX. Расширенное программирование

Отчёт

---

Сергеев Д. О.

7 мая 2025

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Сергеев Даниил Олегович
- Студент
- Направление: Прикладная информатика
- Российский университет дружбы народов
- 1132246837@pfur.ru

## Цель работы

---

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

## Задание

---

Написать командные файлы для задач лабораторной работы

## Ход выполнения лабораторной работы

---



Создадим каталог lab14 с дополнительными директориями для каждого задания. Приступим к выполнению первой задачи.

Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, а дождавшись его освобождения использовать его в течение некоторого времени  $t_2 < t_1$ . Каждая смена состояния должна сопровождаться сообщением. Необходимо запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой, в котором также запущен этот файл, только в привилегированном режиме. Доработаем программу для взаимодействия трёх и более процессов.

В качестве ресурса будем использовать файл `./tmp/resource`. В него будет записываться номер PID. Этот файл будет создаваться после начала использования некоторым процессом. В случае освобождения ресурса файл будет удаляться.

# Выполнение упражнений

```
foot
[dosergeev@vbox 1]$ ./code.sh 7 2 > /dev/pts/1 &
[1] 7953
[dosergeev@vbox 1]$ cat code.sh
[dosergeev@vbox 1]$
[dosergeev@vbox 1]$

foot
[dosergeev@vbox 1]$ ./code.sh 1 5
[PID 7951] waiting for response for 1 seconds...
[PID 7951] 0
[PID 7951] Resource found.
[PID 7951] Using resource for 5 seconds...
[PID 7951] waiting for response for 7 seconds...
[PID 7953] 0
[PID 7953] 1
[PID 7953] 2
[PID 7953] 3
[PID 7953] 4
[PID 7953] 5
[dosergeev@vbox 1]$ [PID 7953] 5
[PID 7953] Resource found.
[PID 7953] Using resource for 2 seconds...
[PID 7953] Releasing resource.
```

Рис. 1: Работа первого скрипта с двумя терминалами.

```
foot
[dosergeev@vbox 1]$ ./code.sh 4 2 > /dev/pts/0 &
[1] 8303
[dosergeev@vbox 1]$

foot
[dosergeev@vbox 1]$ ./code.sh 8 2 > /dev/pts/0 &
[1] 8305
[dosergeev@vbox 1]$

foot
[dosergeev@vbox 1]$ ./code.sh 15 10 > /dev/pts/0 &
[1] 8307
[dosergeev@vbox 1]$

foot
[dosergeev@vbox 1]$ ./code.sh 16 2 > /dev/pts/0 &
[1] 8310
[dosergeev@vbox 1]$

foot
[dosergeev@vbox 1]$ [PID 8301] 0
[PID 8301] Resource found.
[PID 8301] Using resource for 5 seconds...
[PID 8303] waiting for response for 1 seconds...
[PID 8303] 0
[PID 8303] Resource found.
[PID 8303] Using resource for 5 seconds...
[PID 8305] waiting for response for 4 seconds...
[PID 8305] 0
[PID 8307] waiting for response for 15 seconds...
[PID 8307] 0
[PID 8307] 1
[PID 8310] waiting for response for 16 seconds...
[PID 8310] 0
[PID 8305] 1
[PID 8307] 1
[PID 8303] 2
[PID 8305] 2
[PID 8307] 2
[PID 8310] 2
[PID 8305] 3
[PID 8307] 3
[PID 8303] 3
[PID 8305] 4
[PID 8307] 4
[PID 8303] Couldn't get resource for 4 seconds...
[PID 8310] 3
[PID 8305] 5
[PID 8307] 5
[PID 8303] 4
[PID 8305] 6
[PID 8307] 6
[PID 8303] 7
[PID 8310] 7
[PID 8305] Couldn't get resource for 8 seconds...
[PID 8310] 8
[PID 8307] 8
[PID 8310] 9
[PID 8310] 10
[PID 8310] 11
[PID 8310] 12
[PID 8310] 13
[PID 8307] Releasing resource.
[PID 8310] 14
[PID 8310] Resource found.
[PID 8310] Using resource for 2 seconds...
[PID 8310] Releasing resource.
[dosergeev@vbox 1]$
```

Рис. 2: Работа первого скрипта с пятью терминалами.

## Выполнение упражнений

Теперь реализуем команду `man` с помощью командного файла. Используем команду `less` для чтения текстовых файлов, лежащих в архивах каталога `/usr/share/man/man1`. Название команды будет приниматься в качестве аргумента командной строки, а если команды нет, то будет выводиться сообщение об отсутствии справки.

```
if (($#>0))
then
    manual=$(find /usr/share/man/man1/* -name $1.*)
    if [ ${manual:-null} == "null" ]
    then
        echo Command not found
    else
        less -R /usr/share/man/man1/$1.*
    fi
fi
```

Рис. 3: код программы командного файла второго задания.

```
[dosergeev@vbox 2]$ ./code.sh lp1ff1f1  
Command not found  
[dosergeev@vbox 2]$ ./code.sh lp1r34rwa  
Command not found  
[dosergeev@vbox 2]$ ./code.sh getopt
```

Рис. 4: Результат второго скрипта с неизвестной и известной командой.

```
GETOPT(1)                User Commands                GETOPT(1)  
  
NAME  
    getopt - parse command options (enhanced)  
  
SYNOPSIS  
    getopt optstring parameters  
  
    getopt [options] [--] optstring parameters  
  
    getopt [options] -o|--options optstring [options] [--]  
    parameters  
  
DESCRIPTION  
    getopt is used to break up (parse) options in command
```

Рис. 5: Открытая страница справки.

Используя встроенную переменную `$RANDOM`, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Создадим массив с всеми 52 буквами латинского алфавита (заглавными и строчными). С помощью переменной `$RANDOM` будем генерировать размерность последовательности и номер одной из 52 букв.

## Выполнение упражнений

```
size=$((1 + RANDOM % 200))
set -a sequence
alphabet=({a..z} {A..Z})
for ((i=0; i<size; i++))
do
    sym=$((0 + RANDOM % 52))
    sequence[$i]=${alphabet[$sym]}
done
echo ${sequence[*]}
```

Рис. 6: код программы командного файла третьего задания.

```
[dosergeev@vbox 3]$ ./code.sh
P W r g i p H L B g
[dosergeev@vbox 3]$ ./code.sh
I w k d Q u w q Q G L O y V y A n Y
[dosergeev@vbox 3]$ ./code.sh
y m n V j r d O
[dosergeev@vbox 3]$ ./code.sh
r X A o t r d b A w l O V Q y O
[dosergeev@vbox 3]$ ./code.sh
b x E f Q X T
[dosergeev@vbox 3]$ ./code.sh
E y f i v b n l d x Y F N Y i
[dosergeev@vbox 3]$ ./code.sh
r V P e l M
[dosergeev@vbox 3]$ ./code.sh
c o
[dosergeev@vbox 3]$
```

Рис. 7: Результат третьего скрипта с последовательностью до 20 символов.

## Ответы на контрольные вопросы

1. Значения переменной \$1 и строки "exit" написаны слитно с квадратными скобками, из-за чего программа неправильно воспринимает команды.
2. Объединить несколько строк в одну можно с помощью оператора '+=' или с помощью подстановки переменной \${}.

Например:

```
hello="Hello"  
world=" World!"  
hello+= $world
```

*#ИЛИ*

```
echo "${hello}${world}"
```

3. Утилита seq позволяет генерировать последовательности чисел. Её функционал можно реализовать с помощью фигурных скобок или оператора for.

Например:

```
echo {1..10}  
for ((i=1; i<=10; i++)); do echo \"$i; done
```

4. Вычисление выражения  $\$(10/3)$  даст нам целую часть от деления 10 на 3.
5. Основные отличия командной оболочки Zsh от Bash:
- Существует возможность кастомизации;
  - Имеет большое количество плагинов и тем;
  - Имеет подсветку синтаксисов и авто-коррекцию;
  - Имеет более удобную историю команд;



6. `for ((a=1; a <= LIMIT; a++))` – синтаксис верный.
7. По сравнению с другими языками программирования `bash` имеет универсальный способ объявления переменных без указания типов данных (аналогично `python`). Он удобно читается и прост к освоению. В качестве минусов можно выделить нестандартный способ подстановки переменных и высокую чувствительность синтаксиса, как в случае с первым вопросом.

## Вывод

---

В результате выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX/Linux и научился писать более сложные командные файлы.