

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
“РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ”

Факультет физико-математических и естественных наук

ОТЧЕТ

По лабораторной работе №12  
“Программирование в командном процессоре ОС UNIX. Расширенное  
программирование”

Выполнил:

Студент группы: НПИбд-02-21

Студенческий билет: №1032217060

ФИО студента: Королев Адам Маратович

Дата выполнения: 28.05.2022

Москва 2022

## **1 Цель работы:**

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

## 2 Теоретическое введение:

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

## 3 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 имелась возможность взаимодействия трёх и более процессов.

Пишем скрипт.

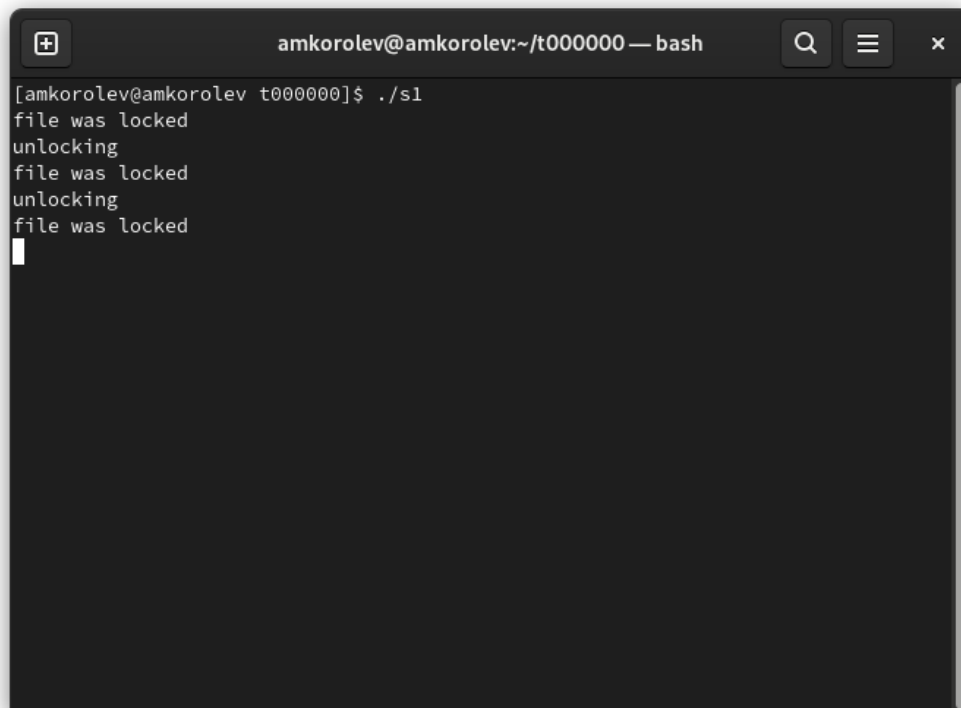


The screenshot shows a terminal window titled 's1' with a path '~/.t000000'. The window contains a shell script that implements a file locking mechanism using the 'flock' command. The script is as follows:

```
1 lockfile="./locking.file"
2
3 exec {fn}>"$lockfile"
4 if test -f "$lockfile"
5 then
6     while [ 1!=0 ]
7     do
8         if flock -n ${fn}
9         then echo "file was locked"
10            sleep 4
11            echo "unlocking"
12            flock -u ${fn}
13
14        else
15            echo "file already locked"
16            sleep 3
17        fi
18    done
19 fi
```

The terminal window has a standard Linux desktop interface with buttons for 'Открыть' (Open), 'Сохранить' (Save), and window controls. The status bar at the bottom indicates the file path, text encoding, tab width, and page information.

Выполняем скрипт

A terminal window with a dark background. The title bar shows 'amkorolev@amkorolev:~/t000000 — bash'. The prompt is '[amkorolev@amkorolev t000000]\$' and the command entered is './s1'. The output of the command is a repeating loop of three lines: 'file was locked', 'unlocking', and 'file was locked'. The cursor is at the end of the last line.

```
[amkorolev@amkorolev:~/t000000 — bash]
[amkorolev@amkorolev t000000]$ ./s1
file was locked
unlocking
file was locked
unlocking
file was locked
█
```

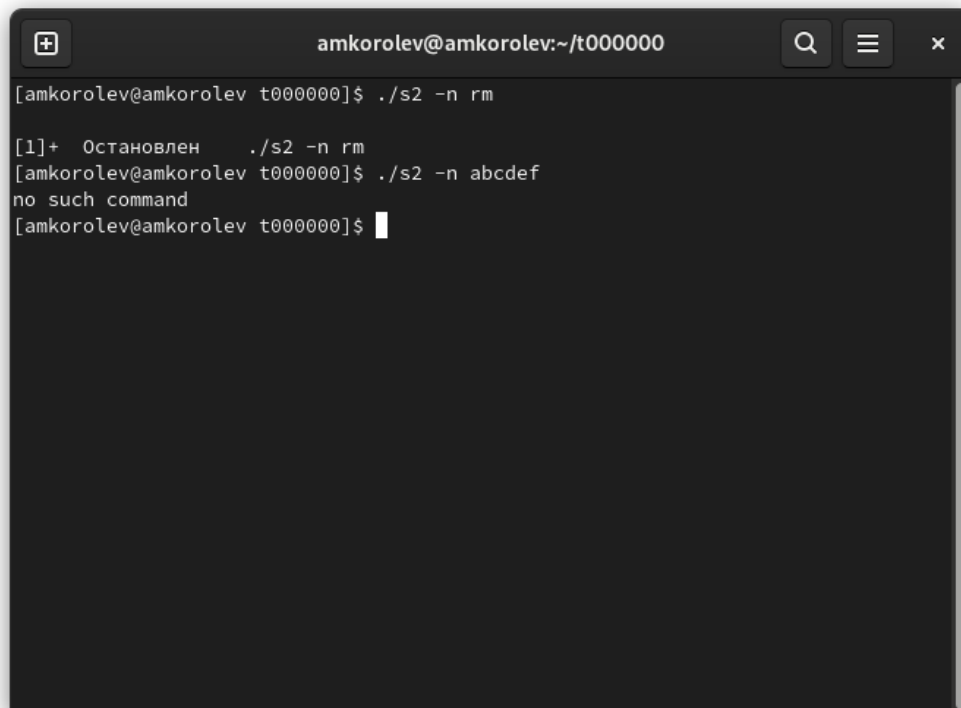
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. 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`.

Пишем скрипт.

```
1 command=""
2
3 while getopts :n: opt
4 do
5 case $opt in
6 n)command="$OPTARG";;
7 esac
8 done
9
10 if test -f "/usr/share/man/man1/$command.1.gz"
11 then less /usr/share/man/man1/$command.1.gz
12 else
13 echo "no such command"
14 fi
```

Сохранение файла «/home/amkorolev/t000000/s2»... Текст ▾ Ширина табуляции: 8 ▾ Стр 14, Стлб 3 ▾ ВСТ

Выполняем скрипт

A terminal window with a dark background. The title bar shows 'amkorolev@amkorolev:~/t000000'. The terminal content shows a sequence of commands and their outputs. The first command is './s2 -n rm', which results in a message '[1]+ Остановлен ./s2 -n rm'. The second command is './s2 -n abcdef', which results in the error 'no such command'. The prompt is ready for the next command.

```
amkorolev@amkorolev:~/t000000
[amkorolev@amkorolev t000000]$ ./s2 -n rm

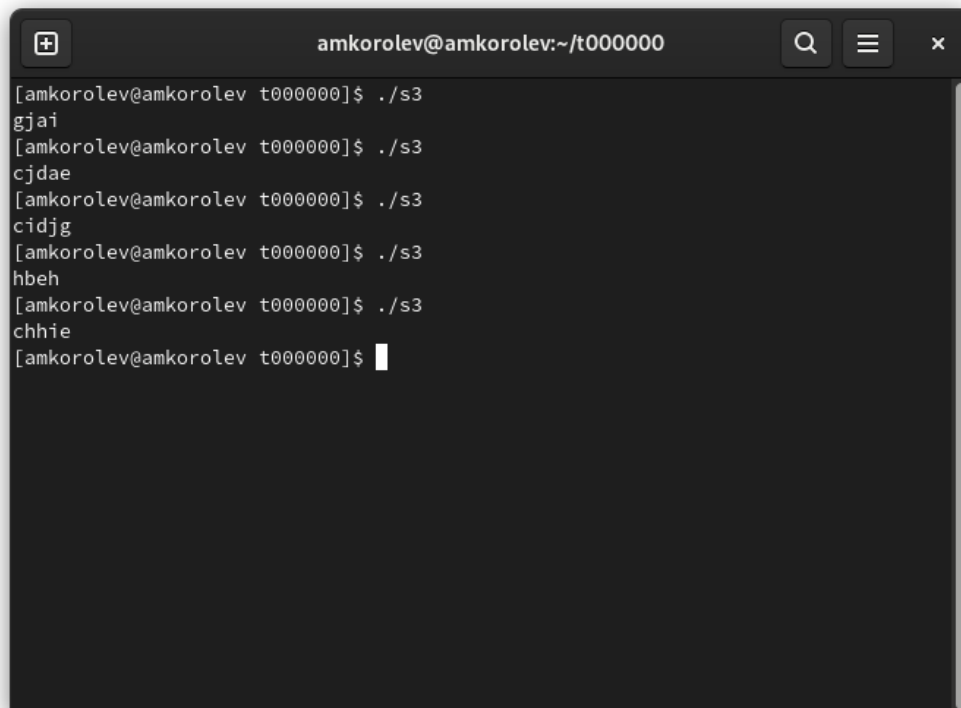
[1]+  Остановлен  ./s2 -n rm
[amkorolev@amkorolev t000000]$ ./s2 -n abcdef
no such command
[amkorolev@amkorolev t000000]$
```

- 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.

Пишем скрипт.



Выполняем скрипт

A terminal window with a dark background and light text. The title bar shows 'amkorolev@amkorolev:~/t000000'. The terminal content shows a series of commands and their outputs. The prompt is '[amkorolev@amkorolev t000000]\$'. The first command is './s3', which outputs 'gjai'. The second command is './s3', which outputs 'cjdae'. The third command is './s3', which outputs 'cidjg'. The fourth command is './s3', which outputs 'hbeh'. The fifth command is './s3', which outputs 'chhie'. The sixth command is './s3', which outputs an empty line. The cursor is at the end of the last line.

```
amkorolev@amkorolev:~/t000000
[amkorolev@amkorolev t000000]$ ./s3
gjai
[amkorolev@amkorolev t000000]$ ./s3
cjdae
[amkorolev@amkorolev t000000]$ ./s3
cidjg
[amkorolev@amkorolev t000000]$ ./s3
hbeh
[amkorolev@amkorolev t000000]$ ./s3
chhie
[amkorolev@amkorolev t000000]$
```

## 4 Выводы:

- В процессе выполнения работы научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 5 Ответы на контрольные вопросы:

1: Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`

\$1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

2: Как объединить (конкатенация) несколько строк в одну?



```
cat file.txt | xargs | sed -e 's/\./\n/g'
```

**3: Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?**

seq - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

**4: Какой результат даст вычисление выражения  $\$((10/3))$ ?**

3

**5: Укажите кратко основные отличия командной оболочки zsh от bash.**

Zsh очень сильно упрощает работу. Но существуют различия. Например, в zsh после `for` обязательно вставлять пробел, нумерация массивов в zsh начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендуется Bash. Если скрипты вам не нужны - Zsh (более простая работа с файлами, например)

**6: Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`**

Верен

**7: Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?**

Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования bash очень сжат. Тот же Си имеет гораздо более широкие возможности для разработчика.