

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

№ 12

Программирование в командном процессоре ОС UNIX. Расширенное
программирование

Коняева Марина Александровна

Содержание

Цель работы	3
Задание	4
Теоретическое введение	5
Выполнение лабораторной работы	6
Выводы	11
Контрольные вопросы	12

Цель работы

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

Задание

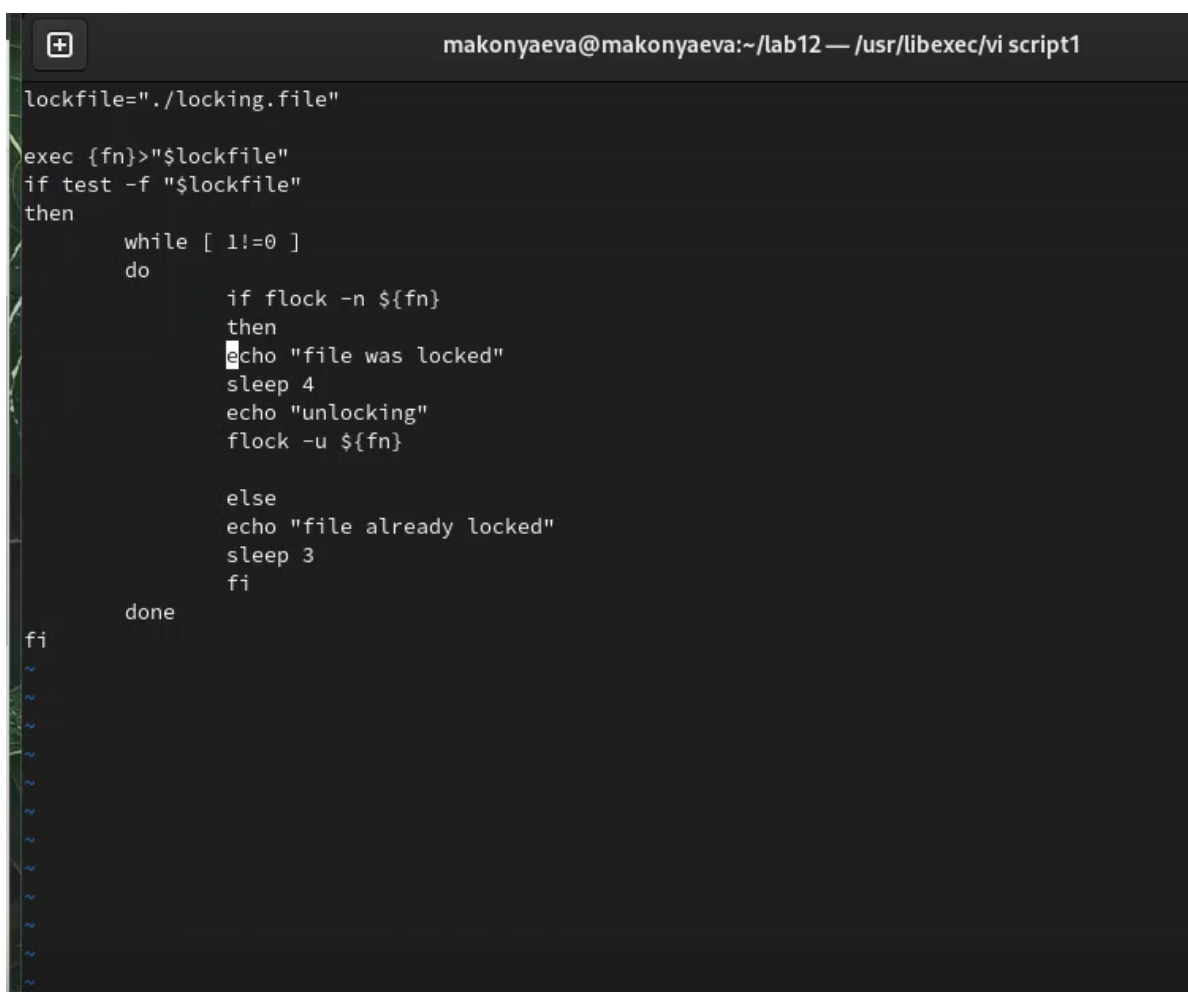
1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию 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.
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`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавит.

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; — С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд; — оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна; — BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation)

Выполнение лабораторной работы

1. Скрипт 1 (изображение 1.1-2)



```
makonyaeva@makonyaeva:~/lab12 — /usr/libexec/vi script1
lockfile="./locking.file"

exec {fn}>"$lockfile"
if test -f "$lockfile"
then
    while [ 1!=0 ]
    do
        if flock -n ${fn}
        then
            echo "file was locked"
            sleep 4
            echo "unlocking"
            flock -u ${fn}

        else
            echo "file already locked"
            sleep 3
        fi
    done
fi
~
~
~
~
~
~
~
~
~
~
```

Изображение 1.1 Скрипт 1

```
[makonyaeva@makonyaeva lab12]$ vi script1
[makonyaeva@makonyaeva lab12]$ vi script1
[makonyaeva@makonyaeva lab12]$ chmod 777 script1
[makonyaeva@makonyaeva lab12]$ ./script1
file was locked
unlocking
file was locked
unlocking
file was locked
unlocking
file was locked
unlocking
file was locked
unlocking
file was locked
aunlocking
file was locked
aunlocking
file was locked
unlocking
file was locked
```

Изображение 1.2 Скрипт 1

2. Скрипт 2 (изображение 2.1-2)

```
command=""

while getopts :n: opt
do
case $opt in
n)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "no such command"
fi
~
~
~
~
~
~
~
~
~
~
```

Изображение 2.1 Скрипт 2


```

ESC[1mNAMEESC[0m
    touch - change file timestamps

ESC[1mSYNOPSISESC[0m
    ESC[1mtouch ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mFILEESC[24m...

ESC[1mDESCRIPTIONESC[0m
    Update the access and modification times of each FILE to the current time.

    A FILE argument that does not exist is created empty, unless ESC[1m-c ESC[22mor ESC[1m-h ESC[22m

    A FILE argument string of - is handled specially and causes touch to change the times of the fi
    ciated with standard output.

    Mandatory arguments to long options are mandatory for short options too.

    ESC[1m-a ESC[22mchange only the access time

    ESC[1m-cESC[22m, ESC[1m--no-createESC[0m
        do not create any files

    ESC[1m-dESC[22m, ESC[1m--dateESC[22m=ESC[4mSTRINGESC[0m
        parse STRING and use it instead of current time

    ESC[1m-f ESC[22m(ignored)

    ESC[1m-hESC[22m, ESC[1m--no-dereferenceESC[0m
        affect each symbolic link instead of any referenced file (useful only on systems that can
        the timestamps of a symlink)

    ESC[1m-m ESC[22mchange only the modification time

    ESC[1m-rESC[22m, ESC[1m--referenceESC[22m=ESC[4mFILEESC[0m
        use this file's times instead of current time

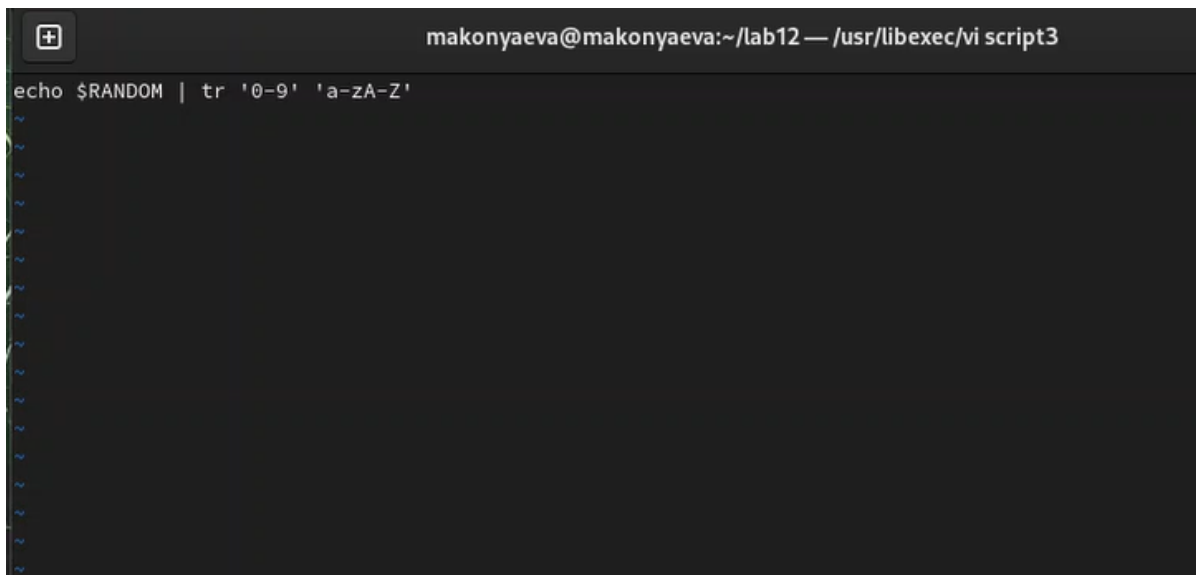
    ESC[1m-t ESC[22mSTAMP
        use [[CC]YY]MMDDhhmm[.ss] instead of current time

    ESC[1m--timeESC[22m=ESC[4mWORDESC[0m
        change the specified time: WORD is access, atime, or use: equ
    modify or

```

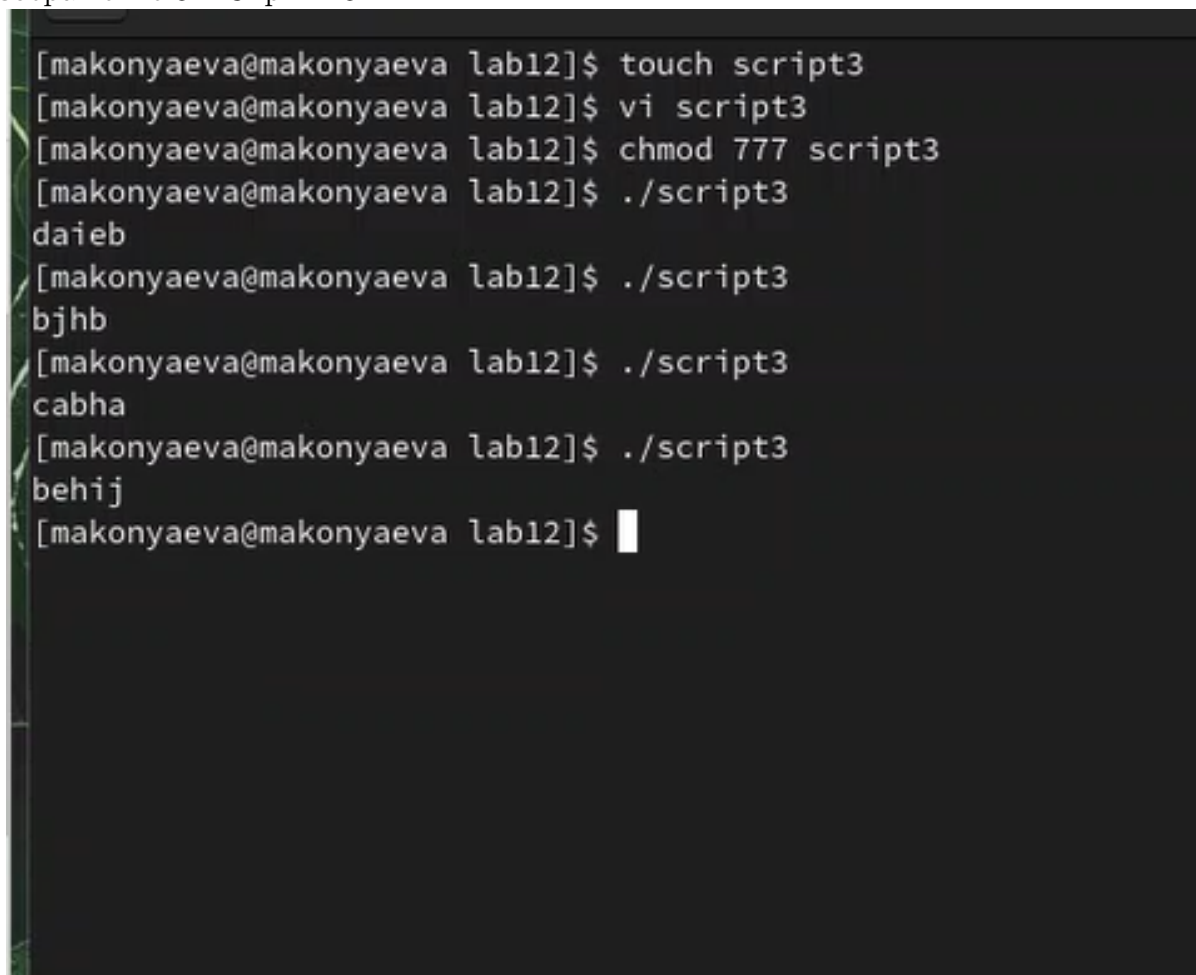
Изображение 2.2 Скрипт 2

3. Скрипт 3 (изображение 3.1-2)

A terminal window with a dark background. The title bar shows the user 'makonyaeva' at host 'makonyaeva' in directory '~/lab12', editing file 'script3' with 'vi'. The terminal content shows the command 'echo \$RANDOM | tr '0-9' 'a-zA-Z'' followed by a series of vertical lines representing the output of the script.

```
makonyaeva@makonyaeva:~/lab12 — /usr/libexec/vi script3
echo $RANDOM | tr '0-9' 'a-zA-Z'
```

Изображение 3.1 Скрипт 3

A terminal window showing the execution of script3. The user runs 'touch script3', 'vi script3', 'chmod 777 script3', and then runs the script multiple times, producing random alphanumeric strings.

```
[makonyaeva@makonyaeva lab12]$ touch script3
[makonyaeva@makonyaeva lab12]$ vi script3
[makonyaeva@makonyaeva lab12]$ chmod 777 script3
[makonyaeva@makonyaeva lab12]$ ./script3
daieb
[makonyaeva@makonyaeva lab12]$ ./script3
bjhb
[makonyaeva@makonyaeva lab12]$ ./script3
cabha
[makonyaeva@makonyaeva lab12]$ ./script3
behij
[makonyaeva@makonyaeva lab12]$
```

Изображение 3.2 Скрипт 3

Выводы

В ходе данной лабораторной работы изучили основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов, а также ответили на контрольные вопросы.

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 `C` имеет гораздо более широкие возможности для разработчика.