

# **Лабораторная работа №12**

**Программирование в командном процессоре ОС UNIX. Расширенное  
программирование**

Лебедева Ольга Андреевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Ход работы</b>	<b>7</b>
<b>4</b>	<b>Вывод</b>	<b>11</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>12</b>

## Список иллюстраций

3.1	Создание папки и файла, запуск редактора vi . . . . .	7
3.2	Текст программы1 . . . . .	8
3.3	Работа программы1 . . . . .	8
3.4	Текст программы2 . . . . .	9
3.5	Вызов программы2 . . . . .	9
3.6	Работа программы2 . . . . .	9
3.7	Текст программы3 . . . . .	10
3.8	Работа программы3 . . . . .	10

## Список таблиц

# 1 Цель работы

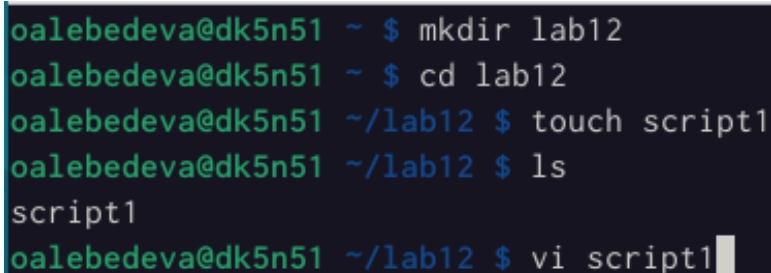
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Теоретическое введение

\$RANDOM – внутренняя функция Bash (не константа), которая возвращает псевдослучайные целые числа в диапазоне 0 - 32767. Функция \$RANDOM не должна использоваться для генерации ключей шифрования.

### 3 Ход работы

1 . Написали 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). Запустили 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. Доработали программу так, чтобы имелась возможность взаимодействия трёх и более процессов. (рис. 3.1) (рис. 3.2) (рис. 3.3)



```
oalebedeva@dk5n51 ~ $ mkdir lab12
oalebedeva@dk5n51 ~ $ cd lab12
oalebedeva@dk5n51 ~/lab12 $ touch script1
oalebedeva@dk5n51 ~/lab12 $ ls
script1
oalebedeva@dk5n51 ~/lab12 $ vi script1
```

Рис. 3.1: Создание папки и файла, запуск редактора vi

```
lockfile="./locking.file"
exec {fn}>"$lockfile"
if test -f "$lockfile"
then
    while [ 1!=0 ]
    do
        if flock -n ${fn}
        then
            echo "the file was locked"
            sleep 5
            echo "unlocking file"
            flock -u ${fn}
        else
            echo "the file already loked"
            sleep 3
        fi
    done
fi
```

Рис. 3.2: Текст программы1

```
oalebedeva@dk5n51 ~/lab12 $ chmod 777 script1
oalebedeva@dk5n51 ~/lab12 $ ./script1
the file was locked
unlocking file
the file already loked
the file already loked
the file already loked
```

Рис. 3.3: Работа программы1

2 . Реализовали команду man с помощью командного файла. Изучили 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. (рис. 3.4) (рис. 3.6) (рис. 3.5)

```
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 command"
fi
```

Рис. 3.4: Текст программы2

```
oalebedeva@dk5n51 ~/lab12 $ ./script2 -n cd
```

Рис. 3.5: Вызов программы2

```
CD(1P)                                POSIX Programmer's Manual                                CD(1P)
PROLOG
This manual page is part of the POSIX Programmer's Manual. The Linux
implementation of this interface may differ (consult the corresponding
Linux manual page for details of Linux behavior), or the interface may
not be implemented on Linux.
NAME
cd - change the working directory
SYNOPSIS
cd [-L|-P] [directory]
```

Рис. 3.6: Работа программы2

3. Используя встроенную переменную \$RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита. Учли, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. 3.7) (рис. 3.8)

```
echo $RANDOM | tr '0-9' 'a-z'
```

Рис. 3.7: Текст программы3

```
oalebedeva@dk5n51 ~/lab12 $ ./script3  
cbiic  
oalebedeva@dk5n51 ~/lab12 $ ./script3  
bbddd
```

Рис. 3.8: Работа программы3

## 4 Вывод

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

## 5 Контрольные вопросы

---

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

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

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 рекомендую `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.