

# **Отчет по лабораторной работе №12**

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

Сагдеров Камал

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Выводы</b>	<b>13</b>
<b>6</b>	<b>Контрольные вопросы</b>	<b>14</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	Текст первой программы . . . . .	10
4.2	Результат . . . . .	10
4.3	Текст второй программы . . . . .	11
4.4	Текст третьей программы . . . . .	11
4.5	Результат . . . . .	12

# 1 Цель работы

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

## 2 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 `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до

32767.

### 3 Теоретическое введение

Оболочка операционной системы (от англ. shell — оболочка) — интерпретатор команд операционной системы (ОС), обеспечивающий интерфейс для взаимодействия пользователя с функциями системы. Зайдя в систему, вы увидите приглашение командной строки — строку, содержащую символ \$ (далее этот символ будет обозначать командную строку). Задача командного интерпретатора состоит в передаче передавать ваши команды операционной системе и прикладным программам, а их ответы — вам. По своим задачам ему соответствует `command.com` в DOS, но функционально оболочки UNIX несравненно богаче. При помощи командных интерпретаторов можно писать небольшие программы — сценарии (скрипты). В Linux доступны следующие командные оболочки:

- Bash — самая распространённая оболочка под Linux. Она ведёт историю команд и предоставляет возможность их редактирования;
- `pdksh` — клон `korn shell`, хорошо известной оболочки в системах UNIX;
- `tcsh` — улучшенная версия `>C shell`;
- `zsh` — новейшая из перечисленных здесь оболочек; реализует улучшенное дополнение и другие удобные функции.

Командный интерпретатор исполняет команды своего языка, заданные в командной строке или поступающие из стандартного ввода или указанного файла.  
**[site:bash?]**

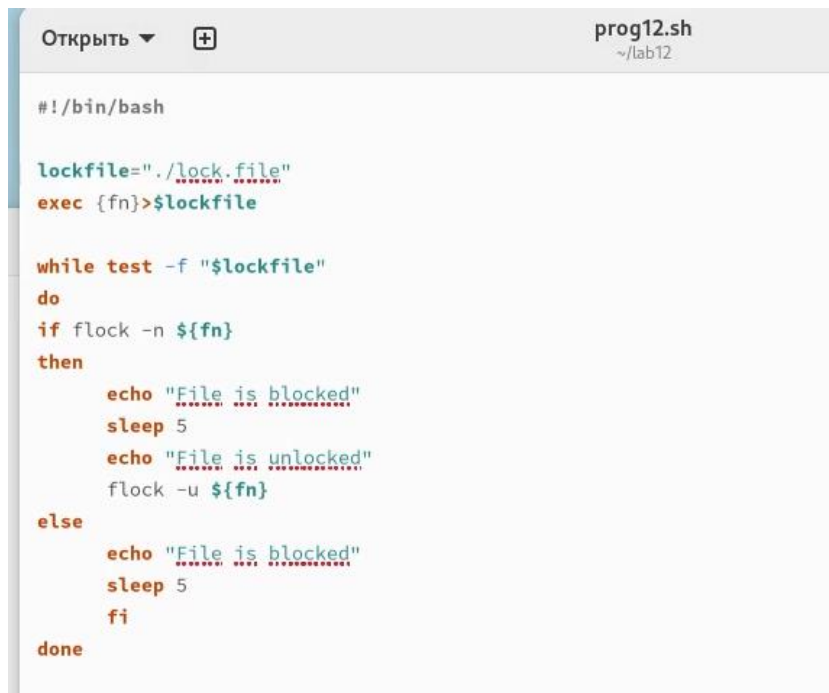
В качестве команд интерпретируются вызовы системных или прикладных утилит, а также управляющие конструкции. Кроме того, оболочка отвечает за раскрытие шаблонов имен файлов и за перенаправление и связывание ввода-вывода утилит.

В совокупности с набором утилит, оболочка представляет собой операционную среду, полноценный язык программирования и мощное средство решения как системных, так и некоторых прикладных задач, в особенности, автоматизации часто выполняемых последовательностей команд [1]



## 4 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 (рис. 4.1), (рис. 4.2).



```
Открыть ▾ + prog12.sh
~/lab12

#!/bin/bash

lockfile="./lock.file"
exec {fn}>$lockfile

while test -f "$lockfile"
do
if flock -n ${fn}
then
echo "File is blocked"
sleep 5
echo "File is unlocked"
flock -u ${fn}
else
echo "File is blocked"
sleep 5
fi
done
```

Рис. 4.1: Текст первой программы



```
ksagderov@fedora:~/lab12

[ksagderov@fedora ~]$ cd lab12
[ksagderov@fedora lab12]$ bash prog12.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
^C
[ksagderov@fedora lab12]$ touch
```

Рис. 4.2: Результат

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` (рис. 4.3).



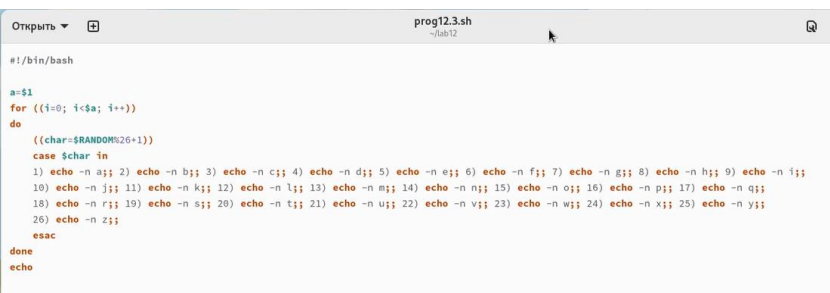
```
Открыть ▾ + prog12.2.sh
~/lab12

#!/bin/bash

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

Рис. 4.3: Текст второй программы

- 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 (рис. 4.4), (рис. 4.5).

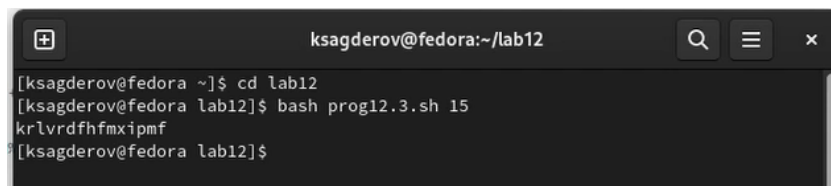


```
Открыть ▾ + prog12.3.sh
~/lab12

#!/bin/bash

a=$1
for ((i=0; i<$a; 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;; 16) echo -n p;; 17) echo -n q;;
18) echo -n r;; 19) echo -n s;; 20) echo -n t;; 21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;;
26) echo -n z;;
esac
done
echo
```

Рис. 4.4: Текст третьей программы

A terminal window with a dark background. The title bar shows 'ksagderov@fedora:~/lab12' and standard window controls. The terminal content shows a sequence of commands and their output: a directory change to 'lab12', running a script 'prog12.3.sh' with argument '15', and the resulting output 'kr1vrdfhfmxi pmf'.

```
ksagderov@fedora:~/lab12
[ksagderov@fedora ~]$ cd lab12
[ksagderov@fedora lab12]$ bash prog12.3.sh 15
kr1vrdfhfmxi pmf
[ksagderov@fedora lab12]$
```

Рис. 4.5: Результат

## 5 Выводы

В процессе выполнения этой лабораторной работы я продолжил осваивать программирование на bash.

## 6 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `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. Как объединить (конкатенация) несколько строк в одну?

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: `VAR1="Hello," VAR2=" World" VAR3="VAR1VAR2" echo "VAR3" : Hello, World : VAR1 = "Hello,"VAR1+ = "World"echo"VAR1"`  
Результат: `Hello, World`

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

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 ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными. seq -w FIRST INCREMENT LAST: эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

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

Результатом данного выражения  $\$((10/3))$  будет 3, потому что это целочисленное деление без остатка.

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

Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cd с помощью Tab В zsh существует калькулятор zcalc, способный выполнять вычисления внутри терминала В zsh поддерживаются числа с плавающей запятой В zsh поддерживаются структуры данных «хэш» В zsh поддерживается раскрытие полного пути на основе неполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim

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

`for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

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

Преимущества скриптового языка bash:

- Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS
- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка bash:
- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языком общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий



# Список литературы

1. Командные оболочки (shells) [Электронный ресурс]. Free Software Foundation. URL: <https://docs.altlinux.org/ru-RU/archive/2.2/html-single/master/install-html/ch06s04.html>.