

Лабораторная работа №12: отчет.

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

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.

Содержание

Цель работы	4
Задание	5
Выполнение лабораторной работы	6
Задание 1	6
Код 1	7
Задание 2	8
Код 2	9
Задание 3	9
Код 3	10
Контрольные вопросы	12
Выводы	15
Список литературы	16

Список иллюстраций

1	Код программы 1	6
2	Результат работы 1	7
3	Код программы 2	8
4	Вызов скрипта	8
5	Результат работы открытие man ls	9
6	Код программы 3	10
7	Результат работы 3	10

Цель работы

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

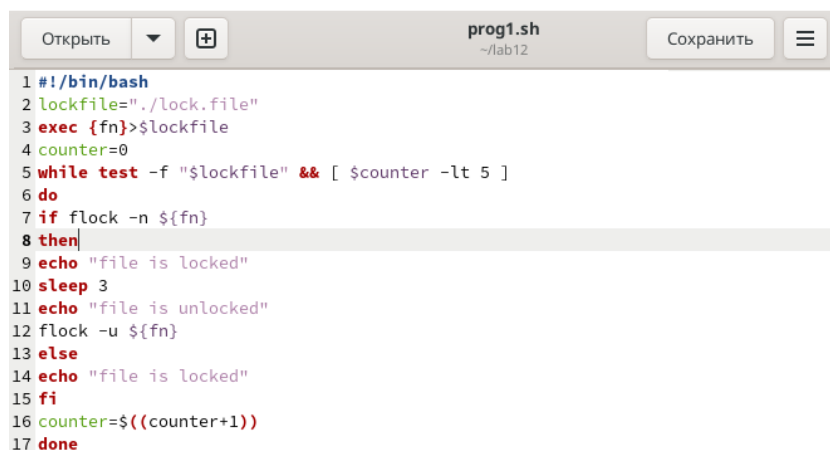
Задание

1. Освоить работу с \$RANDOM.
2. Реализация циклов с задержкой и удержанием.
3. Реализовать свой вариант команды map.

Выполнение лабораторной работы

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 имела возможность взаимодействия трёх и более процессов.



```
1 #!/bin/bash
2 lockfile="./lock.file"
3 exec {fn}>$lockfile
4 counter=0
5 while test -f "$lockfile" && [ $counter -lt 5 ]
6 do
7   if flock -n ${fn}
8   then
9     echo "file is locked"
10    sleep 3
11    echo "file is unlocked"
12    flock -u ${fn}
13  else
14    echo "file is locked"
15  fi
16  counter=$((counter+1))
17 done
```

Рис. 1: Код программы 1

```
max@devolki:~/lab1$ mkdir lab12; cd lab12
max@devolki:~/lab12$ gedit prog1.sh

(gedit:13848): Gtk-WARNING **: 13:29:48.445: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus Error:org.freedesktop.DBus.Error.UnknownMethod: Инт
ерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@devolki:~/lab12$ bash prog1.sh
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
file is locked
file is unlocked
max@devolki:~/lab12$
```

Рис. 2: Результат работы 1

Код 1

```
#!/bin/bash

lockfile="./lock.file"

exec {fn}>$lockfile

counter=0

while test -f "$lockfile" && [ $counter -lt 5 ]
do
if flock -n ${fn}
then
echo "file is locked"
sleep 3
echo "file is unlocked"
flock -u ${fn}
else
echo "file is locked"
fi
counter=$((counter+1))
done
```

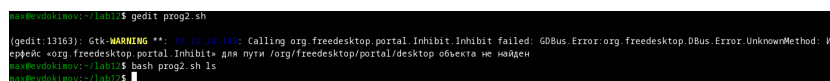
Задание 2

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`.



```
1#!/bin/bash
2a=$1
3if test -f "/usr/share/man/man1/$a.1.gz"
4then less /usr/share/man/man1/$a.1.gz
5else
6echo "There is no such command"
7fi
```

Рис. 3: Код программы 2



```
max@devolki-mov: ~/lab12 $ gedit prog2.sh
[gedit:13163]: Gtk-WARNING **: 17:02:24.440: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Инт
ерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@devolki-mov: ~/lab12 $ bash prog2.sh ls
max@devolki-mov: ~/lab12 $
```

Рис. 4: Вызов скрипта


```
ESC[4mESC[24m(1) User Commands
ESC[4mESC[24m(1)
ESC[1mNAMEESC[0m
ls - list directory contents
ESC[1mSYNOPSISESC[0m
ESC[1mls ESC[22mESC[4mOPTIONESC[24m)... [ESC[4mFILEESC[24m)...
ESC[1mDESCRIPTIONESC[0m
List information about the FILEs (the current directory by default). Sort entries alphabetically if none of ESC[1m-cftuwSUXESC[22mnor ESC[1m--sort
ESC[22mis specified.

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

ESC[1m-aESC[22mESC[1m--allESC[0m
do not ignore entries starting with .

ESC[1m-AESC[22mESC[1m--almost-allESC[0m
do not list implied . and ..

ESC[1m-authorESC[0m
with ESC[1m-lESC[22m, print the author of each file

ESC[1m-bESC[22mESC[1m--escapeESC[0m
print C-style escapes for nongraphic characters

ESC[1m--block-sizeESC[22m=ESC[4mSIZEESC[0m
with ESC[1m-lESC[22m, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

ESC[1m-bESC[22mESC[1m--ignore-backupsESC[0m
do not list implied entries ending with ~

ESC[1mcESC[22mwith ESC[1m-lESC[22m: sort by, and show, ctime (time of last change of file status information); with ESC[1m-lESC[22m: show ctime
/usr/share/man/man1/ls.1.gz
```

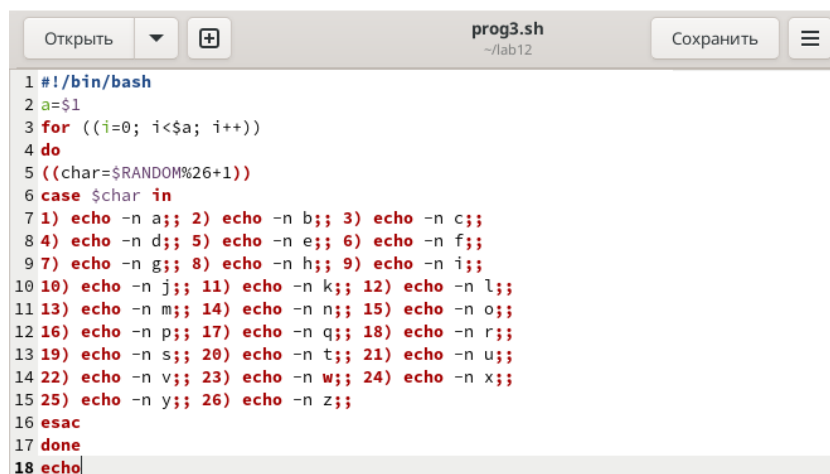
Рис. 5: Результат работы отккрытие man ls

Код 2

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

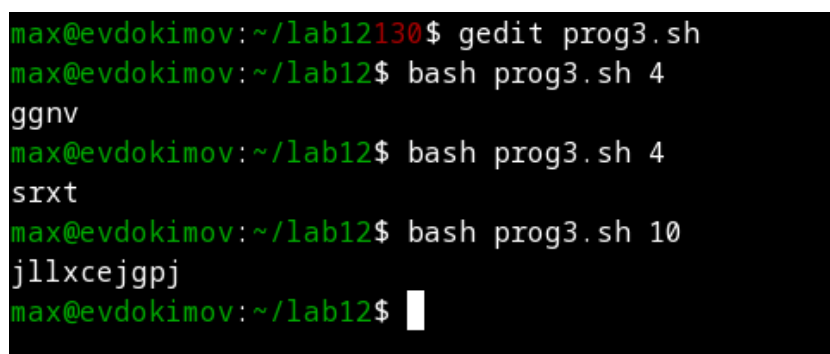
Задание 3

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.



```
1 #!/bin/bash
2 a=$1
3 for ((i=0; i<$a; i++))
4 do
5 ((char=$RANDOM%26+1))
6 case $char in
7 1) echo -n a;; 2) echo -n b;; 3) echo -n c;;
8 4) echo -n d;; 5) echo -n e;; 6) echo -n f;;
9 7) echo -n g;; 8) echo -n h;; 9) echo -n i;;
10 10) echo -n j;; 11) echo -n k;; 12) echo -n l;;
11 13) echo -n m;; 14) echo -n n;; 15) echo -n o;;
12 16) echo -n p;; 17) echo -n q;; 18) echo -n r;;
13 19) echo -n s;; 20) echo -n t;; 21) echo -n u;;
14 22) echo -n v;; 23) echo -n w;; 24) echo -n x;;
15 25) echo -n y;; 26) echo -n z;;
16 esac
17 done
18 echo
```

Рис. 6: Код программы 3



```
max@evdokimov:~/lab12$ gedit prog3.sh
max@evdokimov:~/lab12$ bash prog3.sh 4
ggnv
max@evdokimov:~/lab12$ bash prog3.sh 4
srxt
max@evdokimov:~/lab12$ bash prog3.sh 10
jllxcejgpj
max@evdokimov:~/lab12$
```

Рис. 7: Результат работы 3

Код 3

```
#!/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
```

Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке `“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”`

Результат: Hello, World

Второй: `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 и недостатки 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.

Выводы

В ходе выполнения лабораторной работы были закреплены навыки по работе с скриптами `bash` и изучили некоторые новые функции как `RANDOM`.

Список литературы

1. Лабораторная работа №12