

Шаблон отчёта по лабораторной работе № 13

оп

АДОЛЕ ФЕЙТ

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	15
5	Ответы на контрольные вопросы	16

List of Tables

List of Figures

3.1	Командный файл	9
3.2	Командный файл	10
3.3	Командный файл	11
3.4	Командный файл	12
3.5	Командный файл	12
3.6	Командный файл	13
3.7	Командный файл	14

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 Выполнение лабораторной работы

1. Осуществили вход в систему, создали текстовый документ, затем перешли в него. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t_1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует 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, в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработали программу так, чтобы имелась возможность взаимодействия трёх и более процессов.


```
Файл  Правка  Вид  Закладки  Настройка  Справка
feadole@dk6n63 ~ $ touch lab14.sh
feadole@dk6n63 ~ $ mcedit lab14.sh

feadole@dk6n63 ~ $ chmod +x lab14.sh
feadole@dk6n63 ~ $ ./lab14.sh
lock
work
work
work
work
work
work
work
work
work
feadole@dk6n63 ~ $
```

Figure 3.1: Командный файл

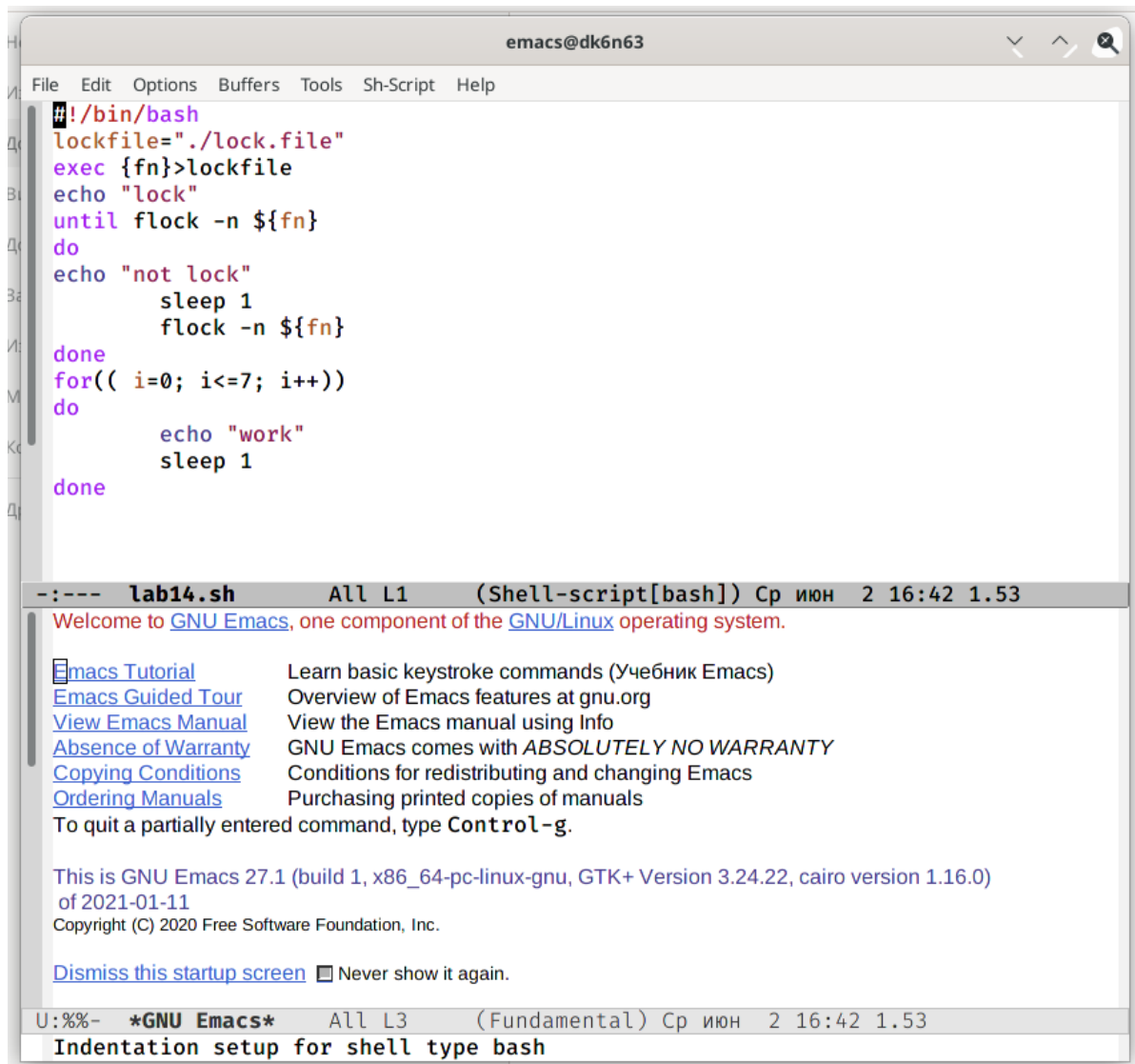


Figure 3.2: Командный файл

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`.

```
man1:lab15.sh — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
zvid(1)          VBI proxy daemon          zvid(1)

NAME
    zvid - VBI proxy daemon

SYNOPSIS
    zvid [ options ]

DESCRIPTION
    zvid is a proxy for VBI devices, i.e. it forwards one or more VBI
    data streams to one or more connected clients and manages channel
    change requests.

OPTIONS
    -dev path
        Path of a device from which to read data. This argument can
        be given several times with different devices.

    -buffers count
        Number of buffers to allocate for capturing VBI raw data
        from devices which support streaming (currently only
        video4linux, rev. 2) A higher number of buffers can prevent
        data loss in case of high latency. The downside is higher
        memory consumption (typically 65kB per buffer.) Default
        count is 8, maximum is 32.

    -nodetach
        Daemon process remains connected to the terminal from which
        it was started (e.g. so that you can stop it by pressing
        Control-C keys). Intended for trouble shooting only.

    -kill
        Terminates a proxy daemon running for the given device.

    -debug level
        Enables debug output: 0= off(default); 1= general messages;
        In addition 2, 4, 8, ... can be added to enable debug output
        for various categories.

    -syslog level
        Enables syslog output.

    -loglevel level
        Log file level

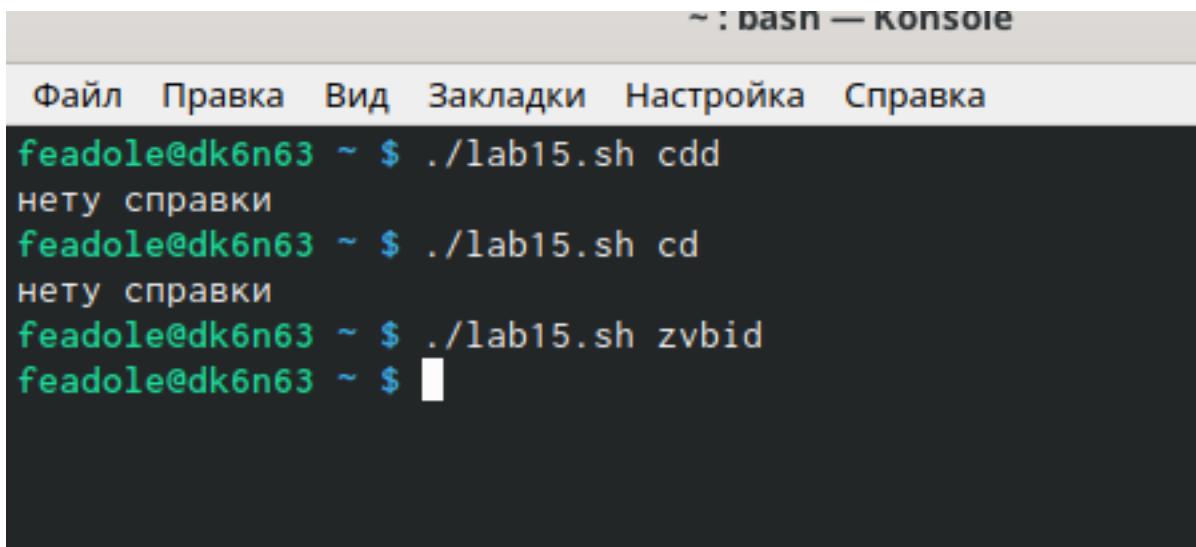
    -logfile path
        Path to the log file.

    -maxclients count
        Max. number of clients which are allowed to connect simulta-
        neously.

    -help
        Print a short description of all command line options.

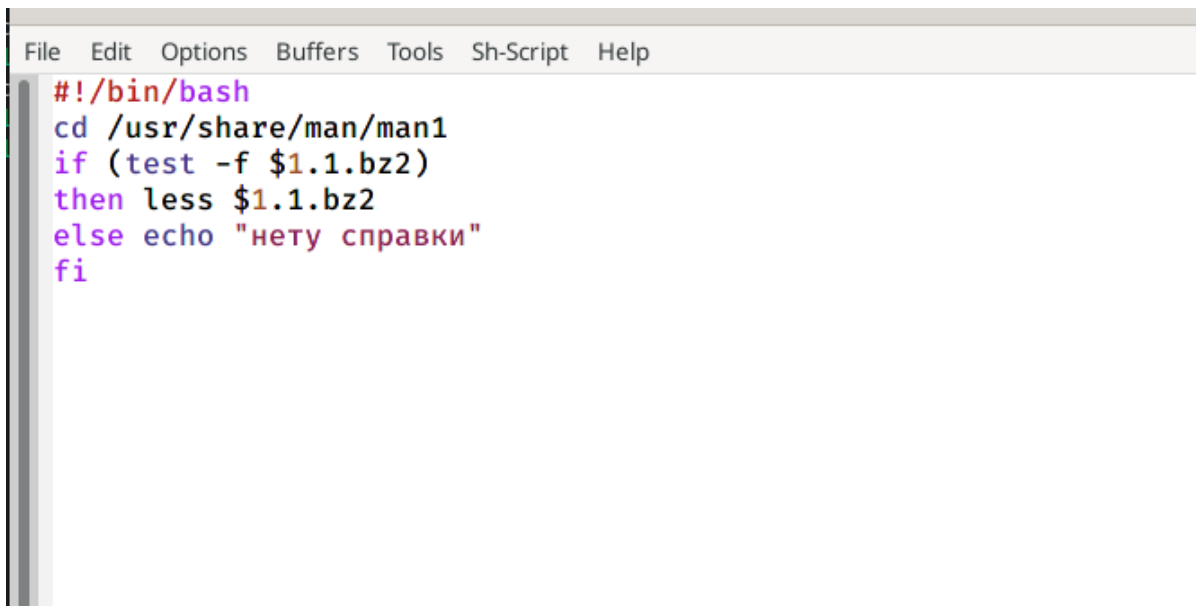
SEE ALSO
    zvbi-chains(1), v4l-conf(8)
    http://zapping.sourceforge.net/ (homepage)
zvbid.1.bz2 lines 1-56
```

Figure 3.3: Командный файл



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
feadole@dk6n63 ~ $ ./lab15.sh cdd
нету справки
feadole@dk6n63 ~ $ ./lab15.sh cd
нету справки
feadole@dk6n63 ~ $ ./lab15.sh zvbid
feadole@dk6n63 ~ $
```

Figure 3.4: Командный файл



```
File  Edit  Options  Buffers  Tools  Sh-Script  Help
#!/bin/bash
cd /usr/share/man/man1
if (test -f $1.1.bz2)
then less $1.1.bz2
else echo "нету справки"
fi
```

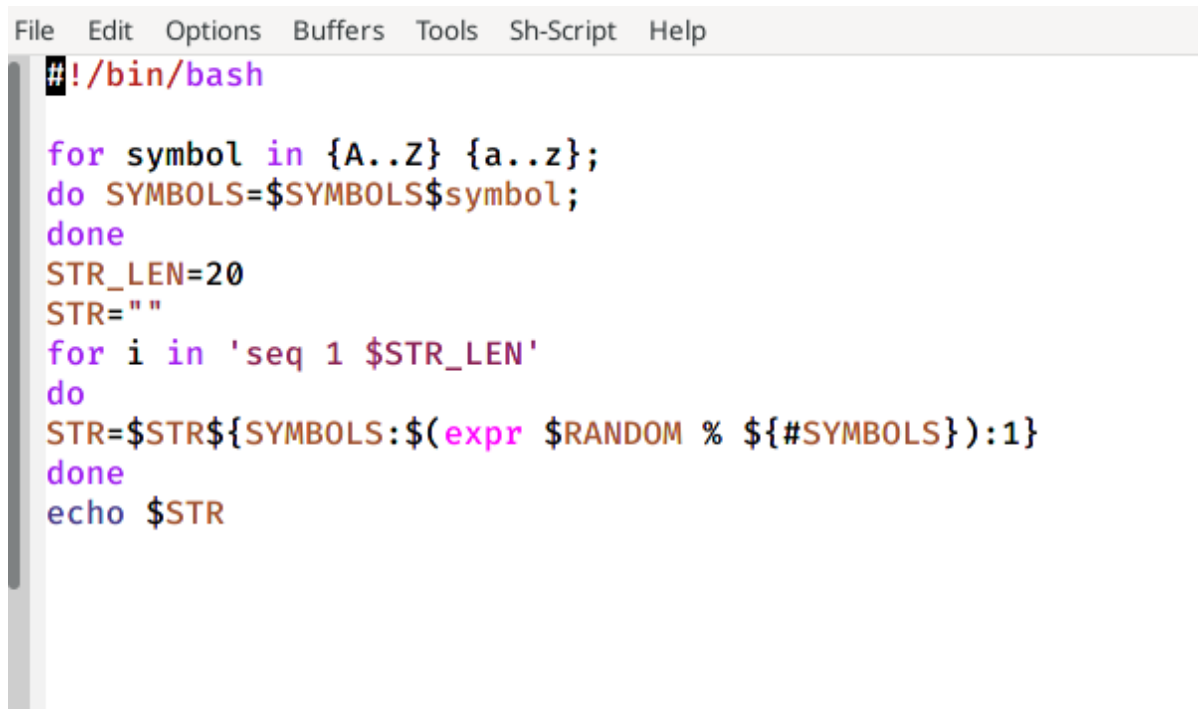
Figure 3.5: Командный файл

- Используя встроенную переменную \$RANDOM, напиши командный файл, генерирующий случайную последовательность букв латинского алфавита.

```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
feadole@dk6n63 ~ $ touch lab16.sh
feadole@dk6n63 ~ $ mcedit lab16.sh

feadole@dk6n63 ~ $ chmod +x lab16.sh
feadole@dk6n63 ~ $ ./lab16.sh
A
h
feadole@dk6n63 ~ $ ./lab16.sh
g
Q
v
feadole@dk6n63 ~ $ ./lab16.sh
O
D
feadole@dk6n63 ~ $ ./lab16.sh
c
p
feadole@dk6n63 ~ $ ./lab16.sh
e
B
feadole@dk6n63 ~ $ ./lab16.sh
e
J
feadole@dk6n63 ~ $ ./lab16.sh
L
feadole@dk6n63 ~ $ ./lab16.sh
z
T
feadole@dk6n63 ~ $ ./lab16.sh
C
x
feadole@dk6n63 ~ $ ./lab16.sh
J
F
feadole@dk6n63 ~ $ ./lab16.sh
m
C
feadole@dk6n63 ~ $ ./lab16.sh
U
r
feadole@dk6n63 ~ $
```

Figure 3.6: Командный файл



```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash

for symbol in {A..Z} {a..z};
do SYMBOLS=$SYMBOLS$symbol;
done
STR_LEN=20
STR=""
for i in `seq 1 $STR_LEN`
do
STR=$STR${SYMBOLS:$(expr $RANDOM % ${#SYMBOLS}):1}
done
echo $STR
```

Figure 3.7: Командный файл

4 Выводы

Мы изучили основы программирования в оболочке ОС UNIX, а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Ответы на контрольные вопросы

1. В строке `while [$1 != "exit"]` \$1 следует внести в кавычки («»).
`while ["$1" != "exit"]`
2. С помощью знака `>|` можно объединить несколько строк в одну.
`>| echo "line 1" "line 2" "line 3"`
3. Эта утилита выводит последовательность целых чисел с заданным шагом.
Также можно реализовать с помощью утилиты `jot`.
`jot 10 1 3`
4. Результатом вычисления выражения `$((10/3))` будет число 3.
`$((10/3))`
5. В `zsh` можно настроить отдельные сочетания клавиш так, как вам нравится.
Использование истории команд в `zsh` ничем особенным не отличается от `bash`. `Zsh` очень удобен для повседневной работы и делает добрую половину рутины за вас. Но стоит обратить внимание на различия между этими двумя оболочками. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1, чего совершенно невозможно понять. Так, если вы используете `shell` для повседневной работы, исключаяющей написание скриптов, используйте `zsh`. Если вам часто приходится писать свои скрипты, только `bash`! Впрочем, можно комбинировать.
6. Синтаксис конструкции `for ((a=1; a <= LIMIT; a++))` верен.
7. Язык `bash` и другие языки программирования:
 - а. Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на `си/си++`, скомпилированных с максимальной оптимизацией;

- b. Скорость работы виртуальной ява-машины с байт-кодом часто превосходит скорость аппаратуры с кодами, получаемыми трансляторами с языков высокого уровня. Ява-машина уступает по скорости только ассемблеру и лучшим оптимизирующим трансляторам;
- c. Скорость компиляции и исполнения программ на яваскрипт в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и превосходит даже некоторые качественные компиляторы, безусловно намного (более чем в 10 раз) обгоняя большинство трансляторов других языков сценариев и подобных им по скорости исполнения программ;
- d. Скорость кодов, генерируемых компилятором языка си фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM;
- e. Скорость ассемблерных кодов x86-64 может меньше, чем аналогичных кодов x86, примерно на 10%;
- f. Оптимизация кодов лучше работает на процессоре Intel;
- g. Скорость исполнения на процессоре Intel была почти всегда выше, за исключением языков лисп, эрланг, аук (gawk, mawk) и бэш. Разница в скорости по бэш скорее всего вызвана разными настройками окружения на тестируемых системах, а не собственно транслятором или железом. Преимущество Intel особенно заметно на 32-разрядных кодах;
- h. Стек большинства тестируемых языков, в частности, ява и яваскрипт, поддерживают только очень ограниченное число рекурсивных вызовов. Некоторые трансляторы (gcc, icc, ...) позволяют увеличить размер стека изменением переменных среды исполнения или параметром;
- i. В рассматриваемых версиях gawk, php, perl, bash реализован динамический стек, позволяющий использовать всю память компьютера. Но perl и, особенно, bash используют стек настолько экстенсивно, что 8-16 ГБ не хватает для расчета ask(5,2,3)