

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: Операционные системы

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

Подготовила: Голощапова И.Б.

Группа: НФИбд_01-20

Цель работы

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

Библиография

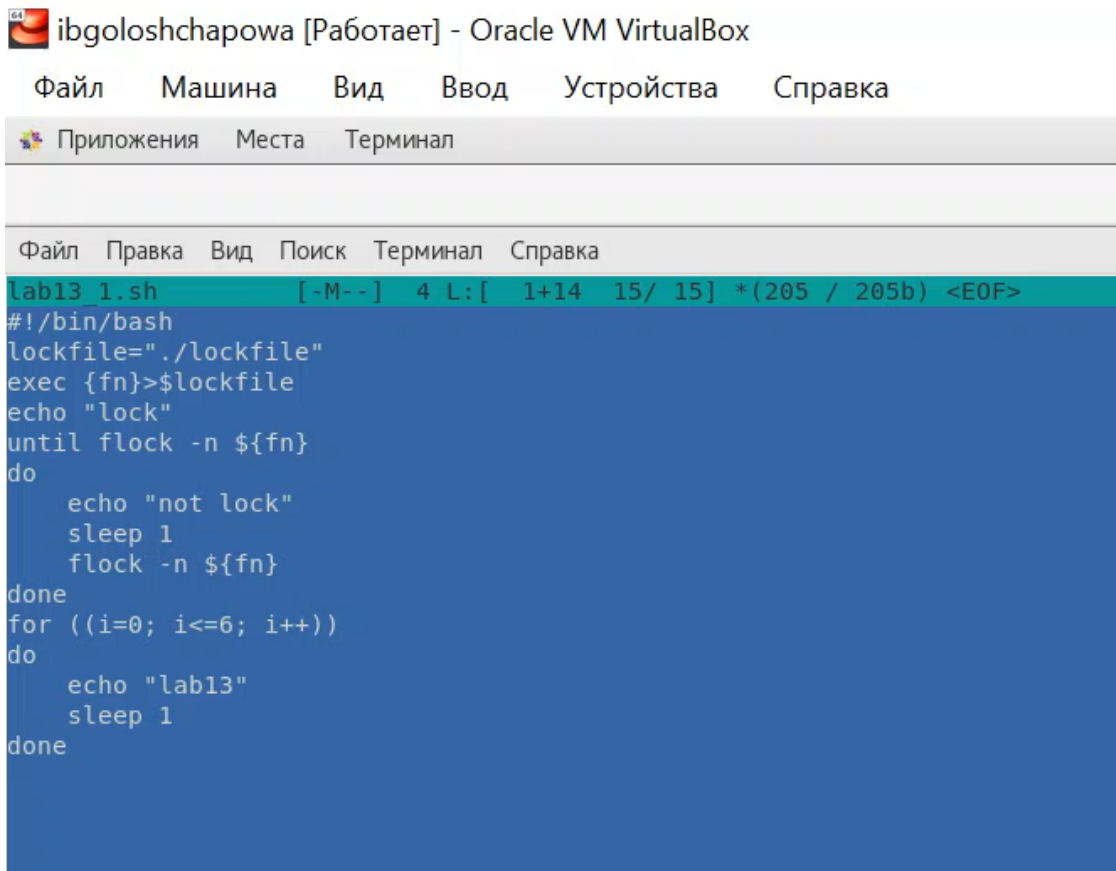
[Командные файлы в Linux](#)

[Википедия - Командная оболочка](#)

[Программные процессоры](#)

Выполнение лабораторной работы

1. Написала 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). Запустила 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. Доработала программу так, чтобы имелась возможность взаимодействия трёх и более процессов.



ibgoloshchapowa [Работает] - Oracle VM VirtualBox

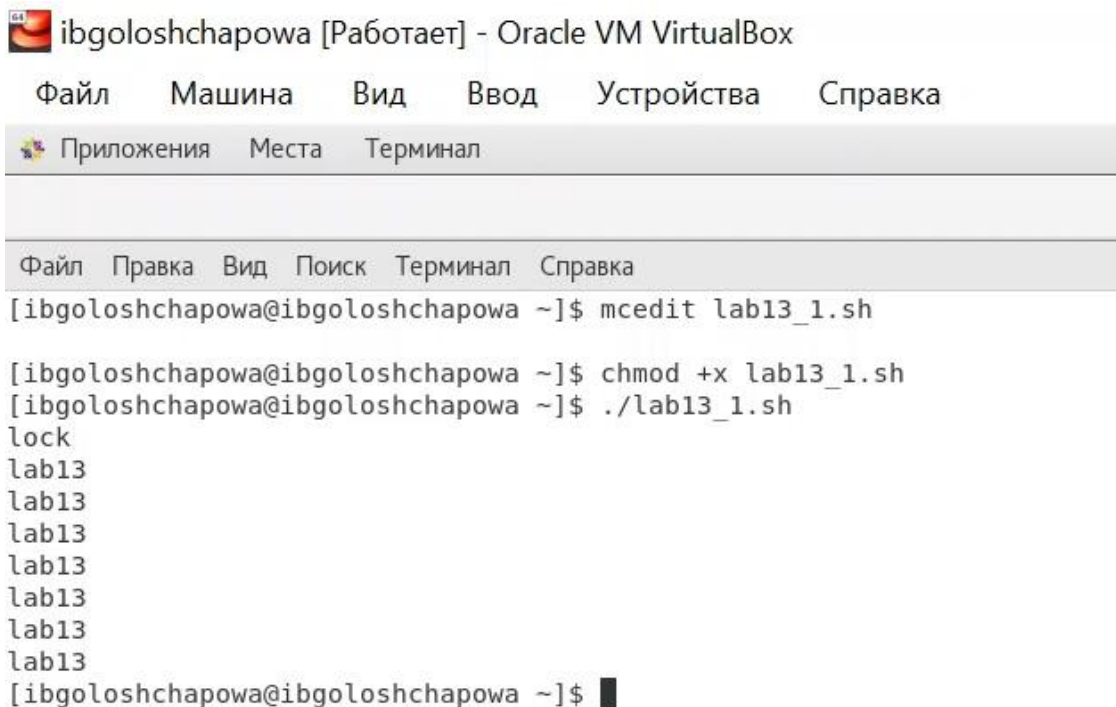
Файл Машина Вид Ввод Устройства Справка

Приложения Места Терминал

Файл Правка Вид Поиск Терминал Справка

```
lab13_1.sh [-M--] 4 L:[ 1+14 15/ 15] *(205 / 205b) <EOF>
#!/bin/bash
lockfile="./lockfile"
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
    echo "not lock"
    sleep 1
    flock -n ${fn}
done
for ((i=0; i<=6; i++))
do
    echo "lab13"
    sleep 1
done
```

Рис.1 "Код программы (1)"



ibgoloshchapowa [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

Приложения Места Терминал

Файл Правка Вид Поиск Терминал Справка

```
[ibgoloshchapowa@ibgoloshchapowa ~]$ mcedit lab13_1.sh

[ibgoloshchapowa@ibgoloshchapowa ~]$ chmod +x lab13_1.sh
[ibgoloshchapowa@ibgoloshchapowa ~]$ ./lab13_1.sh
lock
lab13
lab13
lab13
lab13
lab13
lab13
lab13
[ibgoloshchapowa@ibgoloshchapowa ~]$
```

Рис.2 "Запуск программы (1)"

2. Реализовала команду man с помощью командного файла. Изучила 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.

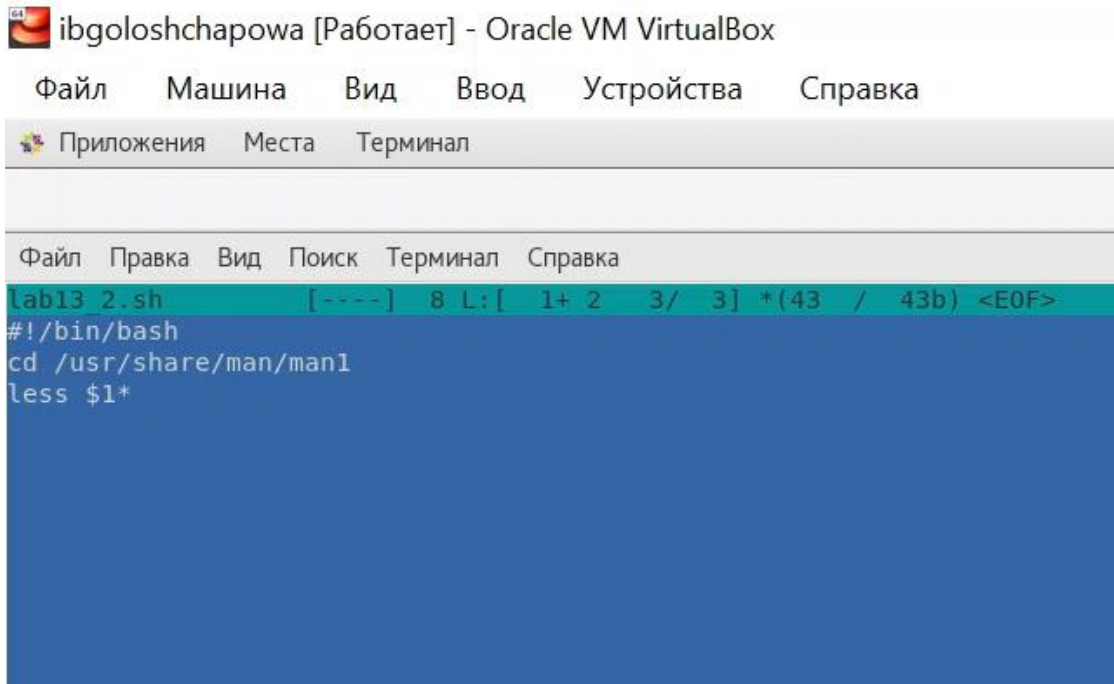


Рис.3 "Код программы (2)"

```
[ibgoloshchapowa@ibgoloshchapowa ~]$ mcedit lab13_2.sh
[ibgoloshchapowa@ibgoloshchapowa ~]$ chmod +x lab13_2.sh
[ibgoloshchapowa@ibgoloshchapowa ~]$ ./lab13_2.sh less
```

Рис.4 "Запуск программы (2)"

```
ibgoloshchapowa [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Приложения  Места  Терминал
ibgoloshchapowa@ibgoloshchapowa:~
Файл  Правка  Вид  Поиск  Терминал  Справка
LESS(1)                                General Commands Manual                                LESS(1)

ESC[1mNAMEESC[0m
less - opposite of more

ESC[1mSYNOPSISESC[0m
ESC[1mless -?ESC[0m
ESC[1mless --helpESC[0m
ESC[1mless -VESC[0m
ESC[1mless --versionESC[0m
ESC[1mless [-[+]aAbcCdeEffGgIiJkLmMnNqQrRsSuUvVwWx~-]ESC[0m
ESC[1m[-b ESC[4mESC[22mspaceESC[24mESC[1m] [-h ESC[4mESC[22mlinesESC[24mESC[1m] [-j ESC[4mESC[22mlineESC[24mESC[1m] [-k ESC[4mESC[22mkeyESC[24mESC[1m] [-l ESC[4mESC[22mlogfileESC[24mESC[1m] [-p ESC[4mESC[22mpatternESC[24mESC[1m] [-P ESC[4mESC[22mpromptESC[24mESC[1m] [-r ESC[4mESC[22mrepeatESC[24mESC[1m] [-s ESC[4mESC[22mspaceESC[24mESC[1m] [-t ESC[4mESC[22mtagsfileESC[24mESC[1m] [-x ESC[4mESC[22mtabESC[24mESC[1m] [-y ESC[4mESC[22mlinesESC[24mESC[1m] [-# ESC[4mESC[22mshiftESC[24mESC[1m] [+][+]ESC[4mESC[22mcmdESC[24mESC[1m] [--] [ESC[4mESC[22mfilenameESC[24mESC[1m]]]
(See the OPTIONS section for alternate option syntax with long option names.)

ESC[1mDESCRIPTIONESC[0m
ESC[4mLessESC[24m is a program similar to ESC[4mmoreESC[24m (1), but which allows backward movement in the file as well as forward movement. Also, ESC[4mlessESC[24m does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like ESC[4mviESC[24m (1). ESC[4mLessESC[24m uses termcap (or terminfo on some systems), so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with a caret.)

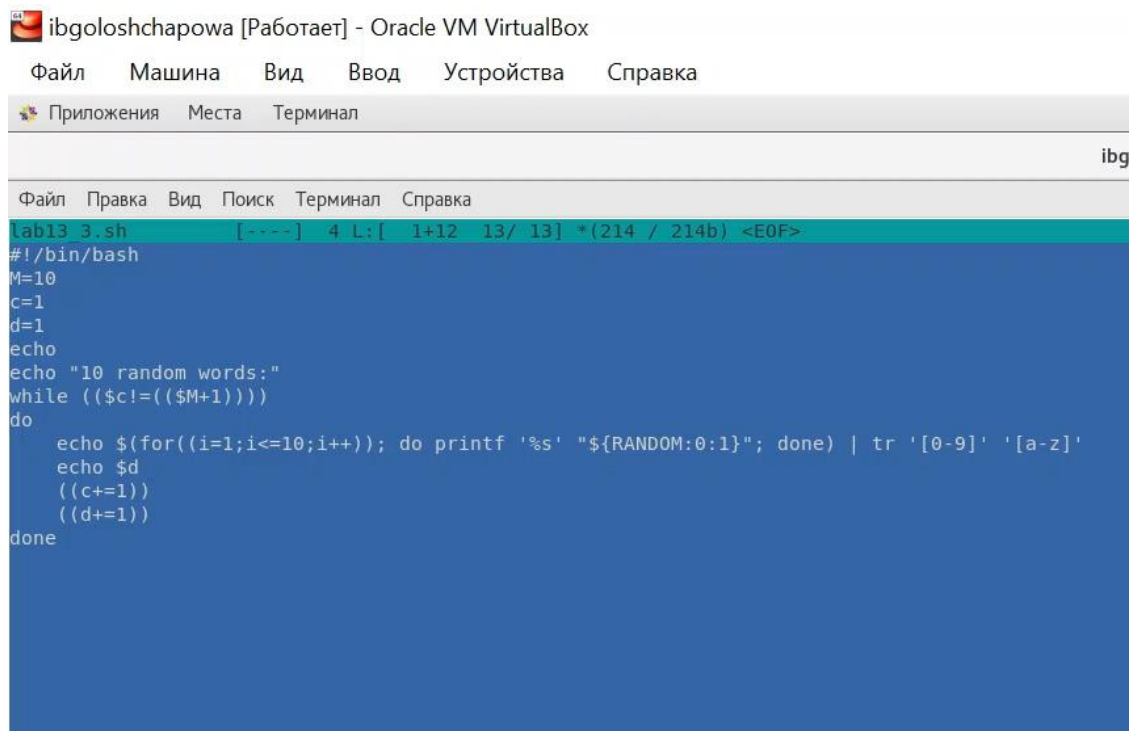
Commands are based on both ESC[4mmoreESC[24m and ESC[4mviESC[24m. Commands may be preceded by a decimal number, called N in the descriptions below. The number is used by some commands, as indicated.

ESC[1mCOMMANDSESC[0m
In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for example ESC-v means the two character sequence "ESCAPE", then "v".

h or H Help: display a summary of these commands. If you forget all the other commands, remember this one.
less: 1 qz (file 1 of 3)
```

Рис.5 "Работа программы (2)"

- Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.



The image shows a screenshot of a terminal window within an Oracle VM VirtualBox environment. The window title is "ibgoloshchapowa [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". Below the menu bar, there is a toolbar with icons for "Приложения", "Места", and "Терминал". The terminal itself has a title bar with "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal content shows a shell script being executed in a bash shell. The script starts with "M=10", "c=1", and "d=1", followed by an "echo" statement. It then enters a "while" loop that continues as long as "c" is not equal to "M+1". Inside the loop, it echoes a string of 10 random words, increments "c" and "d", and loops back. The script ends with "done".

```
lab13 3.sh [----] 4 L: [ 1+12 13/ 13] *(214 / 214b) <EOF>
#!/bin/bash
M=10
c=1
d=1
echo
echo "10 random words:"
while (($c!=($M+1)))
do
    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done) | tr '0-9' '[a-z]'
    echo $d
    ((c+=1))
    ((d+=1))
done
```

Рис.6 "Код программы (3)"

```
ibgoloshchapowa@ibgoloshchapowa ~]$ mcedit lab13_3.sh

ibgoloshchapowa@ibgoloshchapowa ~]$ chmod +x lab13_3.sh
ibgoloshchapowa@ibgoloshchapowa ~]$ ./lab13_3.sh

10 random words:
gfcjccccbb
1
iccbbjccbb
2
degccbibdd
3
ccbcdbcdbcc
4
ccbdcbcdbb
5
bccbebccccc
6
jdbccbiccd
7
bccbidfcdb
8
ecbccfbhde
9
dccccecbccb
10
ibgoloshchapowa@ibgoloshchapowa ~]$
```

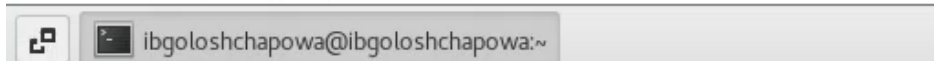


Рис.5 "Работа программы (3)"

Выводы

В ходе лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. В строке `while [$1 != "exit"]` квадратные скобки нужно заменить на круглые.
2. Есть несколько видов конкатенации строк. Например,
`VAR1="Hello,"`
`VAR2=" World"`
`VAR3="\ (VAR1\)VAR2"`

```
echo "$VAR3"
```

3. Команда `seq` выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы. В `bash` можно использовать `seq` с циклом `for`, используя подстановку команд. Например,

```
$ for i in $(seq 1 0.5 4)
do
echo "The number is $i"
done
```

4. Результатом вычисления выражения $\$((10/3))$ будет число 3.

5. Список того, что можно получить, используя `Z Shell` вместо `Bash`:

Встроенная команда `zmv` поможет массово переименовать файлы/директории, например, чтобы добавить `.txt` к имени каждого файла, запустите `zmv -C '(*)(#q.)' '$1.txt'`.

Утилита `zcalc` — это замечательный калькулятор командной строки, удобный способ считать быстро, не покидая терминал.

Команда `zparseopts` — это однострочник, который поможет разобрать сложные варианты, которые предоставляются скрипту.

Команда `autopushd` позволяет делать `popd` после того, как с помощью `cd`, чтобы вернуться в предыдущую директорию.

Поддержка чисел с плавающей точкой (к которой `Bash` не содержит).

Поддержка для структур данных «хэш».

Есть также ряд особенностей, которые присутствуют только в `Bash`:

Опция командной строки `-norc`, которая позволяет пользователю иметь дело с инициализацией командной строки, не читая файл `.bashrc`

Использование опции `-rcfile` с `bash` позволяет исполнять команды из определённого файла.

Отличные возможности вызова (набор опций для командной строки)

Может быть вызвана командой `sh`

`Bash` можно запустить в определённом режиме `POSIX`. Примените `set -o posix`, чтобы включить режим, или `--posix` при запуске.

Можно управлять видом командной строки в `Bash`. Настройка переменной `PROMPT_COMMAND` с одним или более специальными символами настроит её за вас.

`Bash` также можно включить в режиме ограниченной оболочки (с `rbash` или `--restricted`), это означает, что некоторые команды/действия больше не будут доступны:

Настройка и удаление значений служебных переменных SHELL, PATH, ENV, BASH_ENV

Перенаправление вывода с использованием операторов '>', '>|', '<>', '>&', '&>', '>>'

Разбор значений SHELL_OPTS из окружения оболочки при запуске

Использование встроенного оператора exes, чтобы заменить оболочку другой командой

6. Синтаксис конструкции for ((a=1; a <= LIMIT; a++)) верен.

7. Язык bash и другие языки программирования:

-Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на си/си++, скомпилированных с максимальной оптимизацией;

-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;

-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;

-Скорость кодов, генерируемых компилятором языка си фирмы Intel, оказалась заметно меньшей, чем компилятора GNU и иногда LLVM;

-Скорость ассемблерных кодов x86-64 может меньше, чем аналогичных кодов x86, примерно на 10%;

-Оптимизация кодов лучше работает на процессоре Intel;

-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;

-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;

-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 ack(5,2,3)