

Отчет по лабораторной №14

Камалиева Лия Дамировна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
4.1	Контрольные вопросы	16
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	рис.1.1	9
4.2	рис.1.2	10
4.3	рис.1.3	11
4.4	рис.1.4	12
4.5	рис.1.5	13
4.6	рис.1.6	14
4.7	рис.1.7	15
4.8	рис.1.8	16

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 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

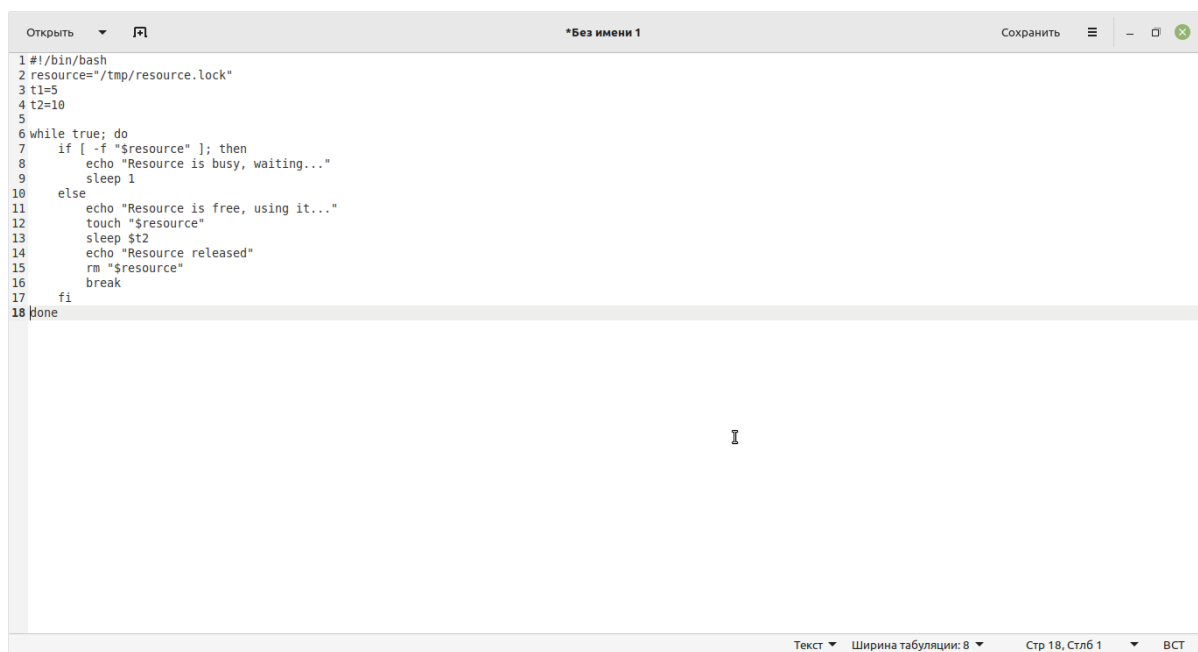
32767

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: – оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation)

4 Выполнение лабораторной работы

Шаг 1.прописываю код по первому заданию и создаю файл semaphore.sh

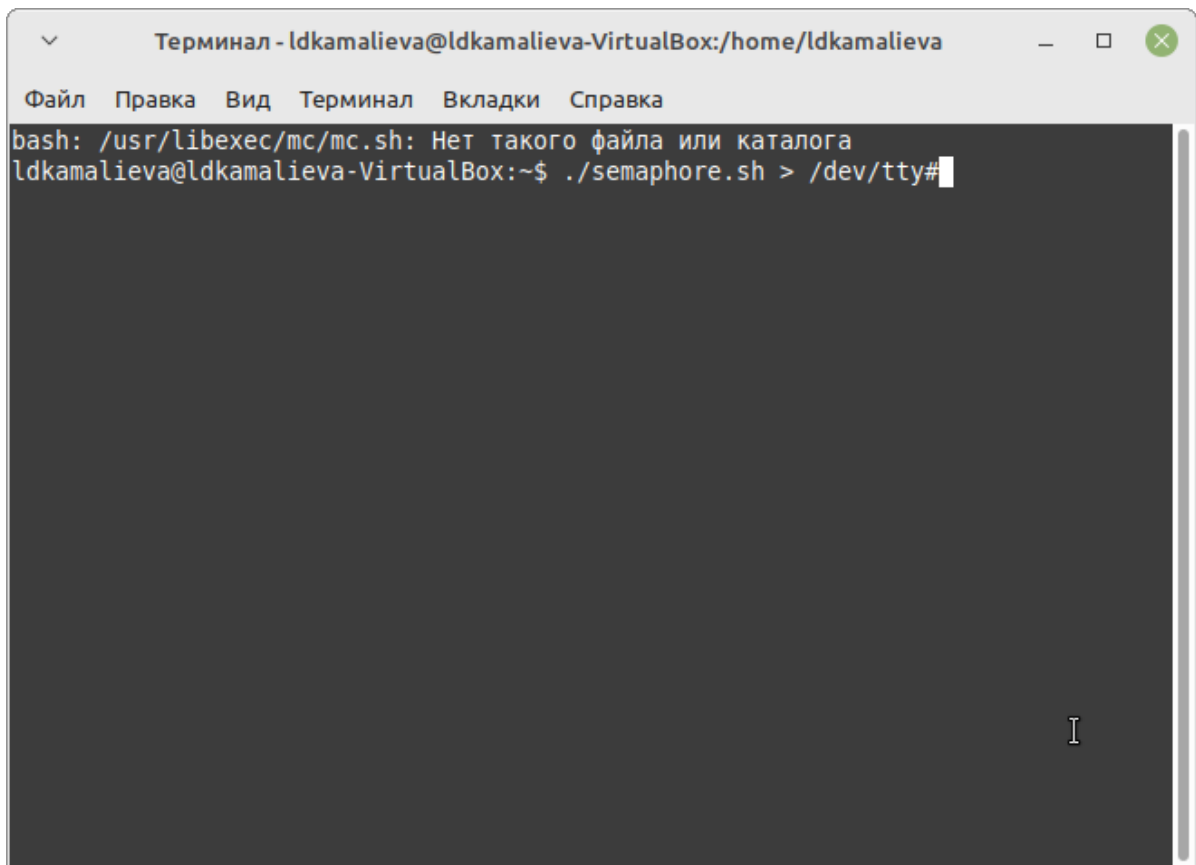


```
1 #!/bin/bash
2 resource="/tmp/resource.lock"
3 t1=5
4 t2=10
5
6 while true; do
7     if [ -f "$resource" ]; then
8         echo "Resource is busy, waiting..."
9         sleep 1
10    else
11        echo "Resource is free, using it..."
12        touch "$resource"
13        sleep $t2
14        echo "Resource released"
15        rm "$resource"
16        break
17    fi
18 done
```

The image shows a code editor window titled "Без имени 1". The editor contains a shell script for a semaphore simulation. The script sets a resource lock file in /tmp, initializes two time variables (t1=5, t2=10), and enters a while loop. Inside the loop, it checks if the resource lock file exists. If it does, it prints "Resource is busy, waiting..." and sleeps for 1 second. If it doesn't exist, it prints "Resource is free, using it...", creates the lock file, sleeps for t2 seconds (10), prints "Resource released", removes the lock file, and breaks the loop. The script ends with a "done" statement. The editor interface includes a menu bar with "Открыть", "Сохранить", and window controls. The status bar at the bottom shows "Текст", "Ширина табуляции: 8", "Стр 18, Стлб 1", and "ВСТ".

Рис. 4.1: рис.1.1

Шаг 2. запускаю код



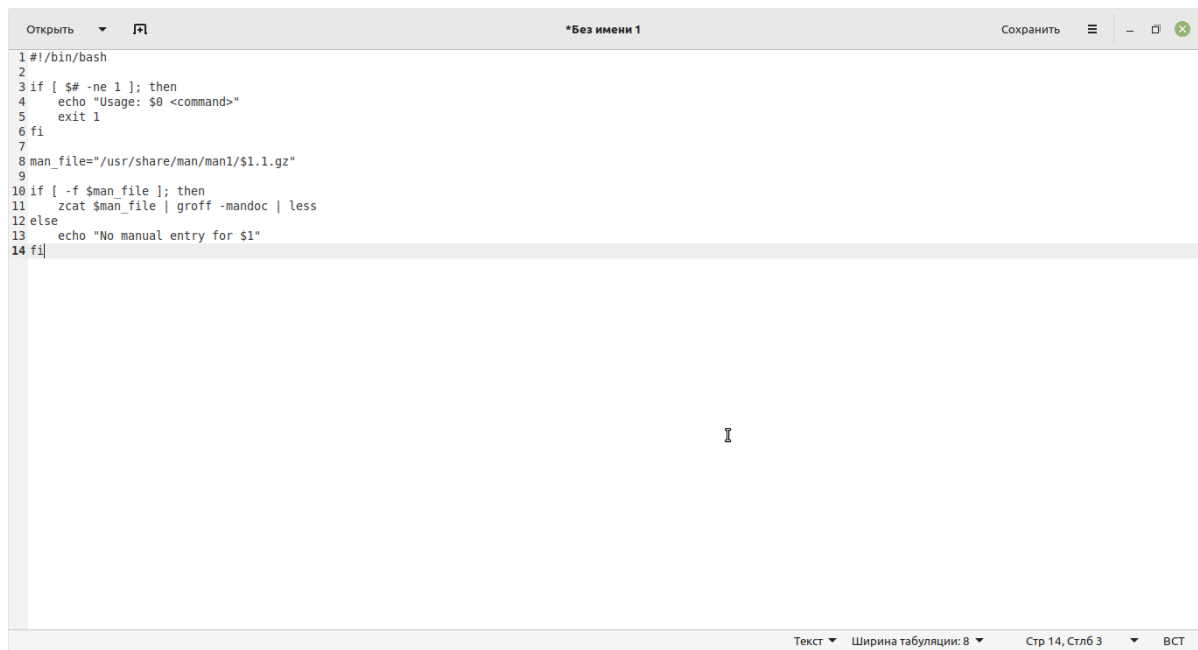
Терминал - ldkamalieva@ldkamalieva-VirtualBox:/home/ldkamalieva

Файл Правка Вид Терминал Вкладки Справка

```
bash: /usr/libexec/nc/nc.sh: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ ./semaphore.sh > /dev/tty#
```

Рис. 4.2: рис.1.2

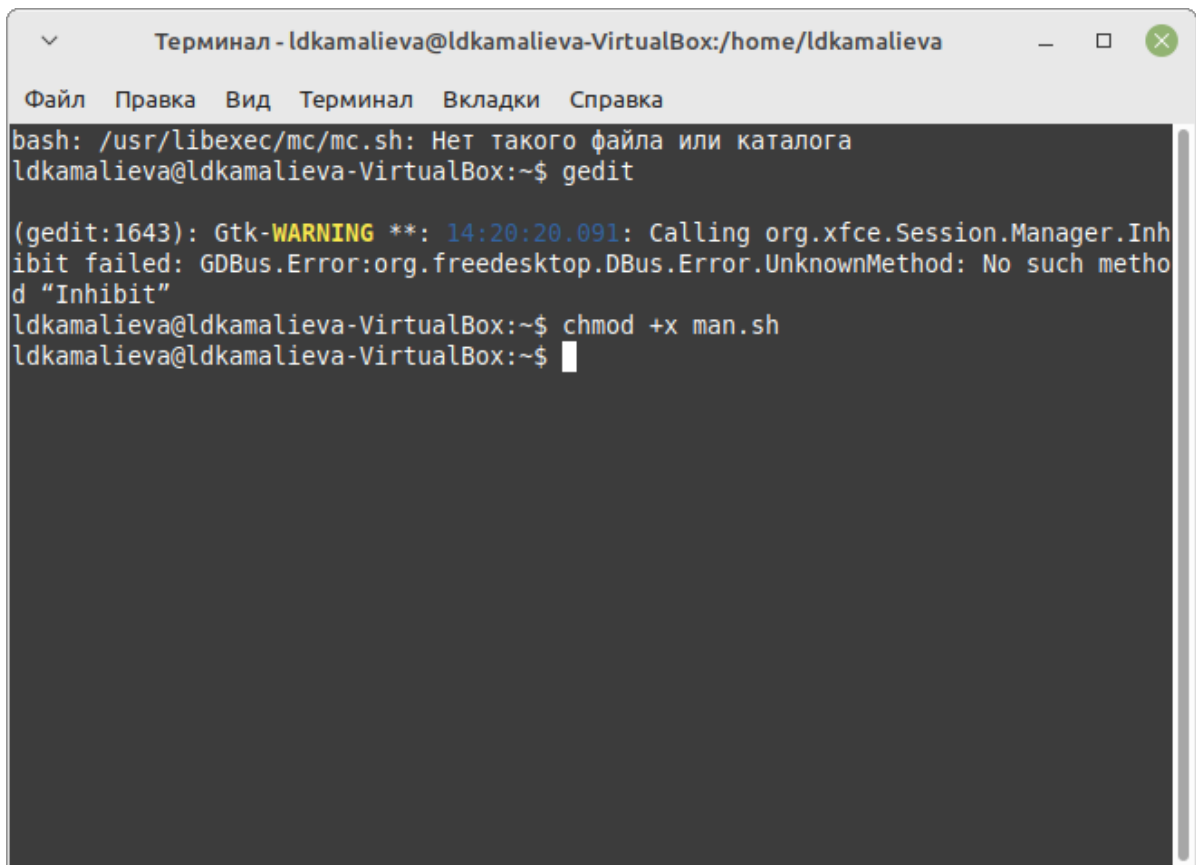
Шаг 3. прописываю код по 2 заданию и сохраняю файл под именем man.sh



```
1#!/bin/bash
2
3if [ $# -ne 1 ]; then
4    echo "Usage: $0 <command>"
5    exit 1
6fi
7
8man_file="/usr/share/man/man1/$1.1.gz"
9
10if [ -f $man_file ]; then
11    zcat $man_file | groff -mandoc | less
12else
13    echo "No manual entry for $1"
14fi
```

Рис. 4.3: рис.1.3

Шаг 4. делаю код исполняемым

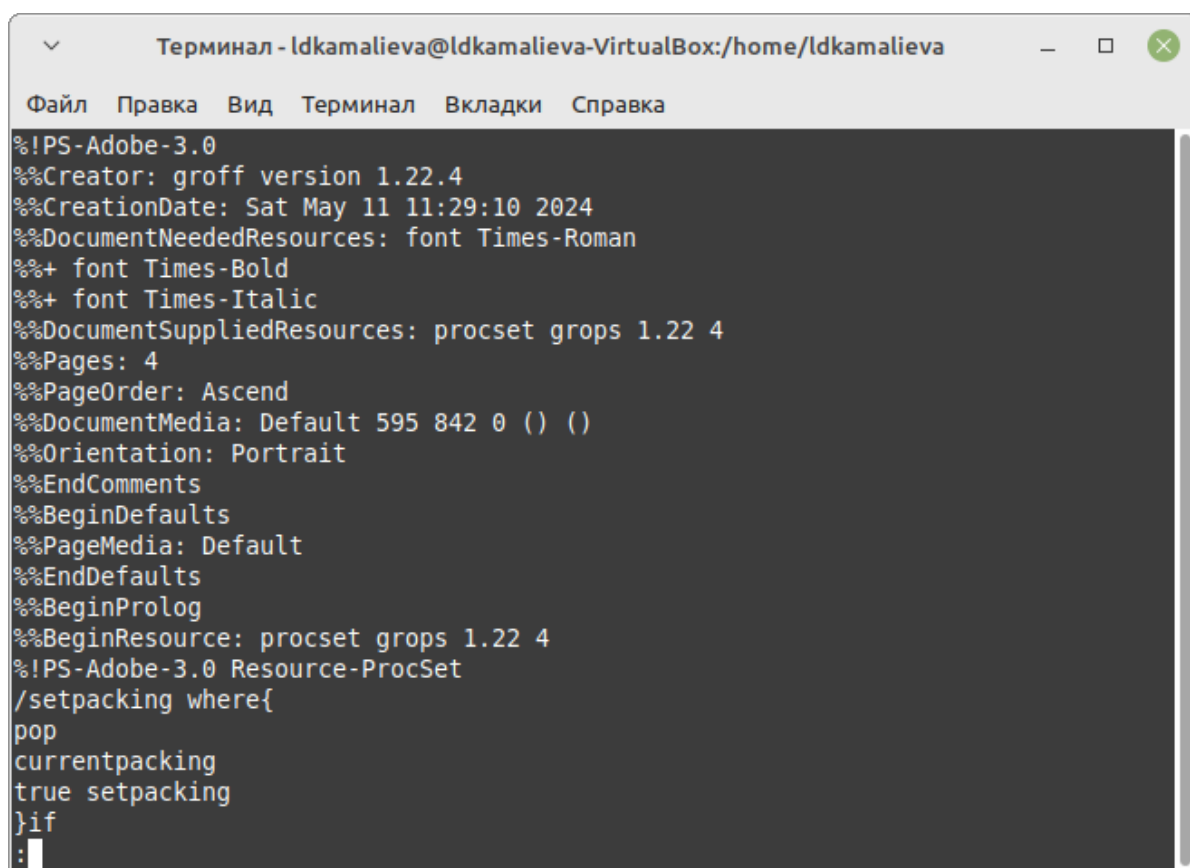


```
Терминал - ldkamalieva@ldkamalieva-VirtualBox:/home/ldkamalieva
Файл  Правка  Вид  Терминал  Вкладки  Справка
bash: /usr/libexec/nc/nc.sh: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ gedit

(gedit:1643): Gtk-WARNING **: 14:20:20.091: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
ldkamalieva@ldkamalieva-VirtualBox:~$ chmod +x man.sh
ldkamalieva@ldkamalieva-VirtualBox:~$
```

Рис. 4.4: рис.1.4

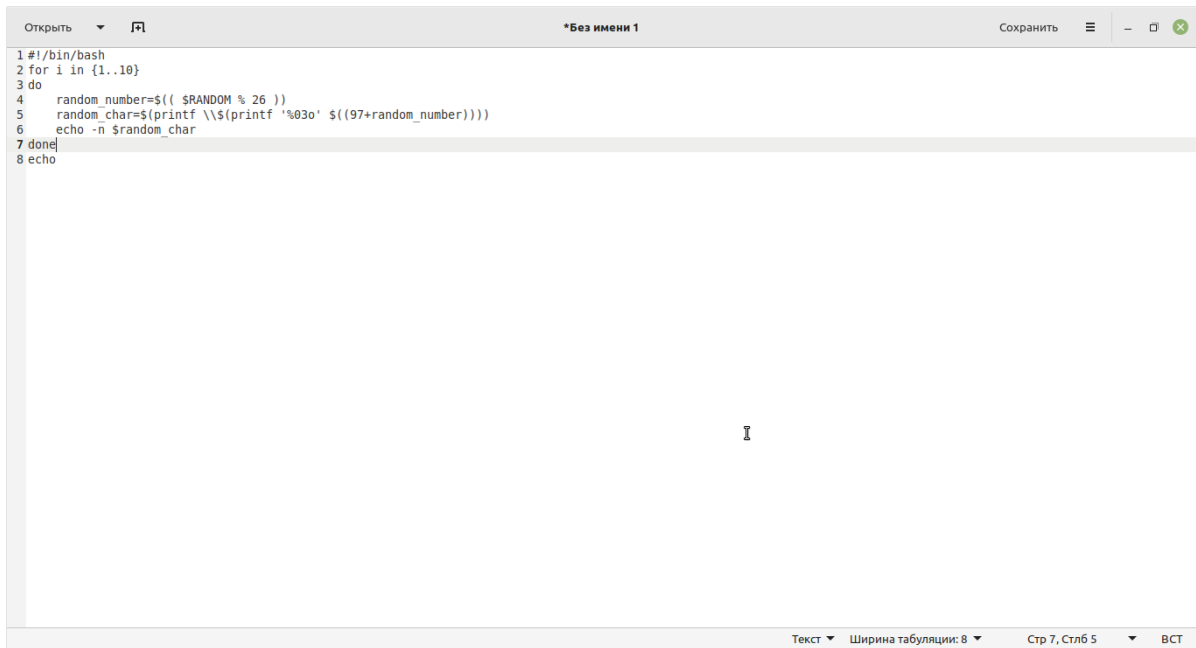
Шаг 5. Этот скрипт будет пытаться найти файл справки для указанной команды и откроет его в less, если он найден. Если файла справки нет, он выдаст сообщение об отсутствии справки.

A screenshot of a terminal window titled "Терминал - ldkamalieva@ldkamalieva-VirtualBox:/home/ldkamalieva". The window has a menu bar with "Файл", "Правка", "Вид", "Терминал", "Вкладки", and "Справка". The terminal displays the following PostScript code:

```
%!PS-Adobe-3.0
%%Creator: groff version 1.22.4
%%CreationDate: Sat May 11 11:29:10 2024
%%DocumentNeededResources: font Times-Roman
%%+ font Times-Bold
%%+ font Times-Italic
%%DocumentSuppliedResources: procset grops 1.22 4
%%Pages: 4
%%PageOrder: Ascend
%%DocumentMedia: Default 595 842 0 () ()
%%Orientation: Portrait
%%EndComments
%%BeginDefaults
%%PageMedia: Default
%%EndDefaults
%%BeginProlog
%%BeginResource: procset grops 1.22 4
%!PS-Adobe-3.0 Resource-ProcSet
/setpacking where{
pop
currentpacking
true setpacking
}if
:
```

Рис. 4.5: рис.1.5

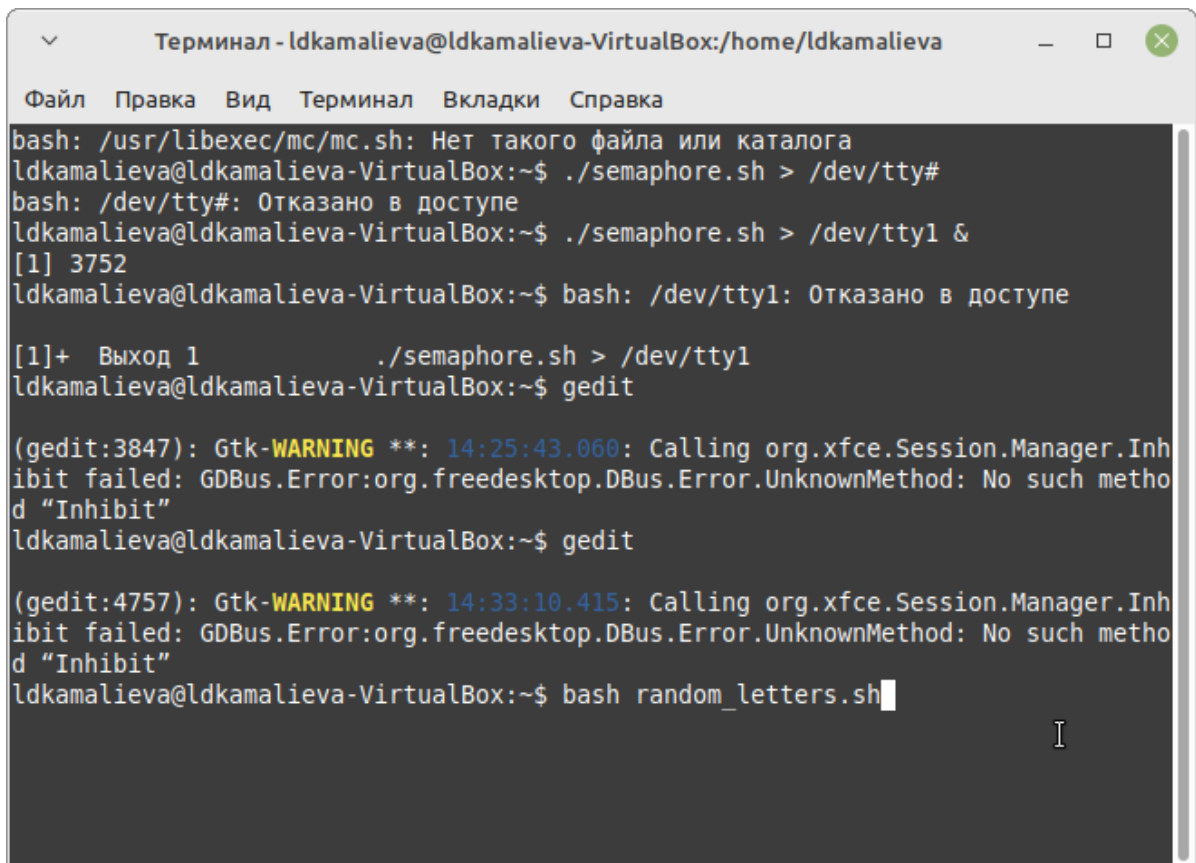
Шаг 6. прописываю скрипт для задания с рандомными числами



```
1#!/bin/bash
2for i in {1..10}
3do
4    random_number=$(( $RANDOM % 26 ))
5    random_char=$(printf \\$(printf '%03o' $((97+random_number))))
6    echo -n $random_char
7done
8echo
```

Рис. 4.6: рис.1.6

Шаг 7. пишу команду, чтобы запустить код



```
Терминал - ldkamalieva@ldkamalieva-VirtualBox:/home/ldkamalieva
Файл  Правка  Вид  Терминал  Вкладки  Справка
bash: /usr/libexec/nc/nc.sh: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ ./semaphore.sh > /dev/tty#
bash: /dev/tty#: Отказано в доступе
ldkamalieva@ldkamalieva-VirtualBox:~$ ./semaphore.sh > /dev/tty1 &
[1] 3752
ldkamalieva@ldkamalieva-VirtualBox:~$ bash: /dev/tty1: Отказано в доступе

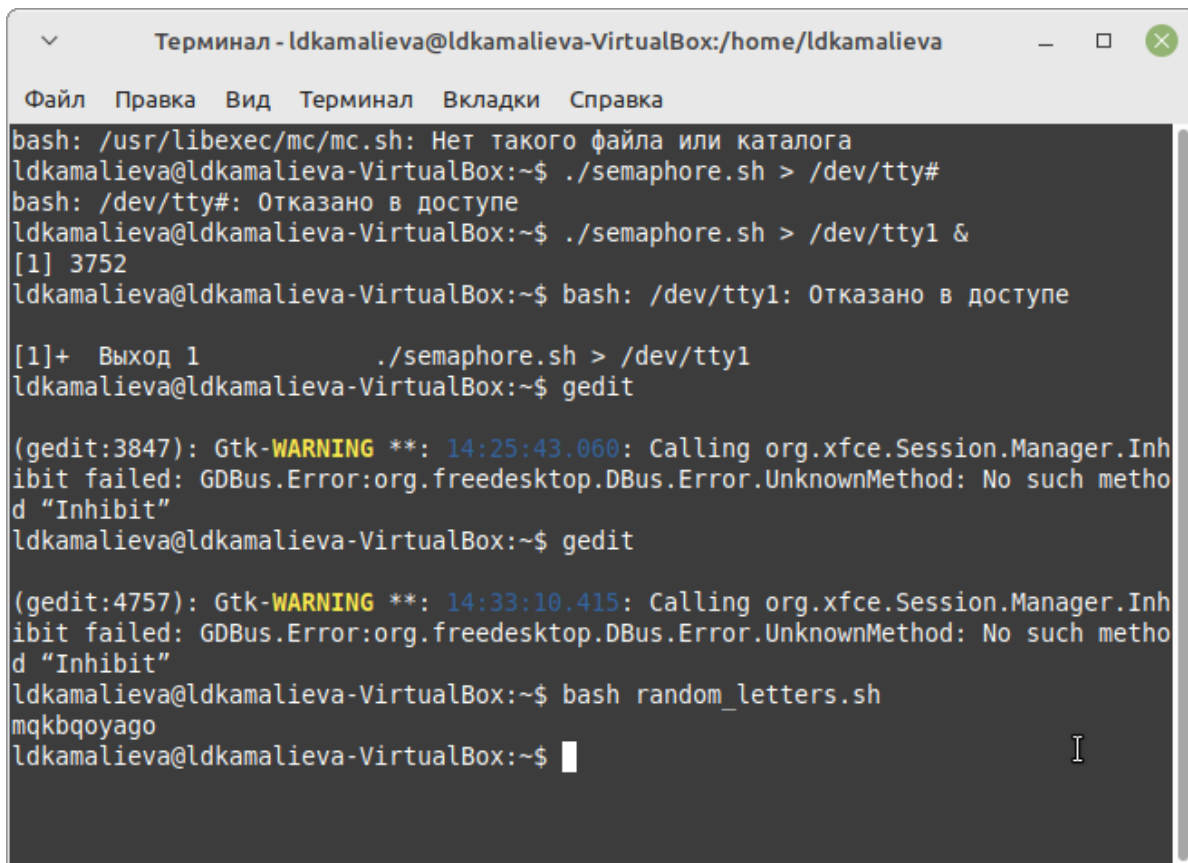
[1]+  Выход 1          ./semaphore.sh > /dev/tty1
ldkamalieva@ldkamalieva-VirtualBox:~$ gedit

(gedit:3847): Gtk-WARNING **: 14:25:43.060: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
ldkamalieva@ldkamalieva-VirtualBox:~$ gedit

(gedit:4757): Gtk-WARNING **: 14:33:10.415: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
ldkamalieva@ldkamalieva-VirtualBox:~$ bash random_letters.sh
```

Рис. 4.7: рис.1.7

Шаг 8. проверяю работу, всё работает



```
Терминал - ldkamalieva@ldkamalieva-VirtualBox:/home/ldkamalieva
Файл  Правка  Вид  Терминал  Вкладки  Справка
bash: /usr/libexec/nc/nc.sh: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ ./semaphore.sh > /dev/tty#
bash: /dev/tty#: Отказано в доступе
ldkamalieva@ldkamalieva-VirtualBox:~$ ./semaphore.sh > /dev/tty1 &
[1] 3752
ldkamalieva@ldkamalieva-VirtualBox:~$ bash: /dev/tty1: Отказано в доступе

[1]+  Выход 1          ./semaphore.sh > /dev/tty1
ldkamalieva@ldkamalieva-VirtualBox:~$ gedit

(gedit:3847): Gtk-WARNING **: 14:25:43.060: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
ldkamalieva@ldkamalieva-VirtualBox:~$ gedit

(gedit:4757): Gtk-WARNING **: 14:33:10.415: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
ldkamalieva@ldkamalieva-VirtualBox:~$ bash random_letters.sh
mqkbqoyago
ldkamalieva@ldkamalieva-VirtualBox:~$
```

Рис. 4.8: рис.1.8

4.1 Контрольные вопросы

1 Каково предназначение команды `getopts`? Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда

getopts может распознать аргумент, то она возвращает истину. Принято включать getopts в цикл while и анализировать введенные данные с помощью оператора case. Функция getopts включает две специальные переменные среды OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента. Функция getopts также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2 Какое отношение метасимволы имеют к генерации имён файлов? При перечислении имён файлов текущего каталога можно использовать следующие символы: 1 соответствует произвольной, в том числе и пустой строке; 2 ? соответствует любому одинарному символу; 3 [c1-c2] соответствует любому символу, лексикографически находящемуся между символами c1 и c2. Например, 1.1 echo выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; 1.2. ls.c выведет все файлы с последними двумя символами, совпадающими с c. 1.3. echoprogram.? выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog.. 1.4.[a-z] соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3 Какие операторы управления действиями вы знаете? Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования bash предоставляет возможность использовать такие управляющие конструкции, как for, case, if и while. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования bash. Поэтому при описании языка программирования bash термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвраща-

ют код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4 Какие операторы используются для прерывания цикла? Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5 Для чего нужны команды `false` и `true`? Следующие две команды `OCUNIX` используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done until false do echo hello mike done`.

6 Что означает строка `if test -f mans/i.$s`, встречаемая в командном файле? Строка `if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернёт нулевое значение (ложь).

7 Объясните различия между конструкциями `while` и `until`. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список команд в

строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

5 Выводы

я ознакомилась с функциями etacs

Список литературы