

Лабораторная работа № 12. Презентация

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

Козлов Всеволод Павлович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	10
4	Ответы на контрольные вопросы	11
	Список литературы	14

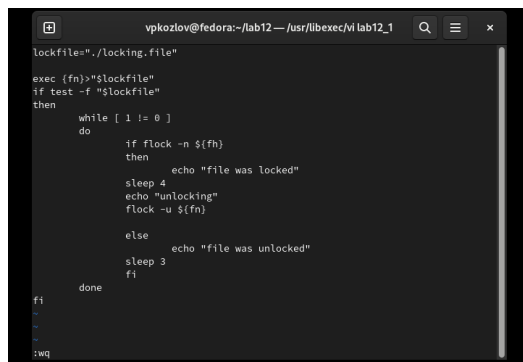
Список иллюстраций

2.1	Создание и запуск файла_1	6
2.2	Код файла_1	7
2.3	Создание и запуск файла_2	7
2.4	Код файла_2	8
2.5	Работа файла_2	8
2.6	Создание и запуск файла_3	9
2.7	Код файла_3	9

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.

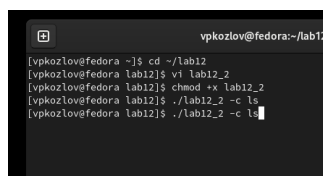


```
lockfile="/locking.file"

exec {fn}>${lockfile}
if test -f "${lockfile}"
then
  while [ 1 != 0 ]
  do
    if flock -n ${fn}
    then
      echo "file was locked"
      sleep 4
      echo "unlocking"
      flock -u ${fn}
    else
      echo "file was unlocked"
      sleep 3
    fi
  done
fi
```

Рис. 2.2: Код файла_1

Создал и запустил командный файл, реализующий команду map (код ниже) (рис. [2.3]).



```
vpkozlov@fedora:~/lab12
[vpkozlov@fedora ~]$ cd ~/lab12
[vpkozlov@fedora lab12]$ vi lab12_2
[vpkozlov@fedora lab12]$ chmod +x lab12_2
[vpkozlov@fedora lab12]$ ./lab12_2 -c ls
[vpkozlov@fedora lab12]$ ./lab12_2 -c ls
```

Рис. 2.3: Создание и запуск файла_2

Код командного файла, реализующего команду map (рис. [2.4]).

```
vpkozlov@fedora:~/lab12 — /usr/libexec/vi lab12_2
command=""
while getopts :c: opt
do
case $opt in
c)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "No such a command:"
fi
fi

:WQ
```

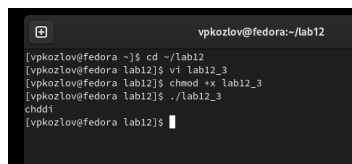
Рис. 2.4: Код файла_2

Демонстрация работы командного файла_2 (рис. [2.5]).

```
vpkozlov@fedora:~/lab12 — bash
LS(1) User Commands LS(1)
ESC[ImNAMEESC[Om
ls - list directory contents
ESC[ImSYNOPSISESC[Om
ESC[Imls ESC[22mESC[4mOPTIONESC[24m)... [ESC[4mFILEESC[24m)...
ESC[ImDESCRIPTIONESC[Om
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of ESC[Im-cftuvSUXESC[22mnor ESC[Im-
-sortESC[22mms speci-
fied.
Mandatory arguments to long options are mandatory for short options
too.
ESC[Im--ESC[22m, ESC[Im--allESC[Om
do not ignore entries starting with .
ESC[Im--ESC[22m, ESC[Im--almost-allESC[Om
do not list implied . and ..
:
```

Рис. 2.5: Работа файла_2

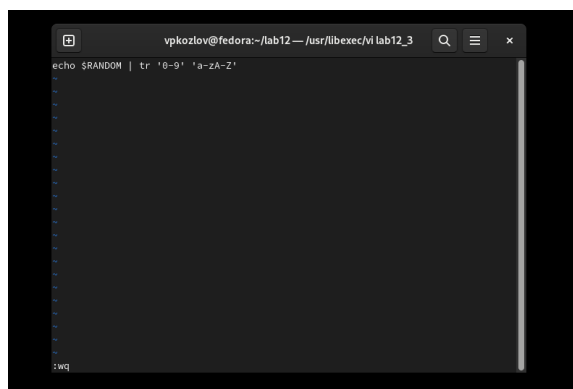
Создал и запустил командный файл, генерирующий случайную последователь-
ность букв латинского алфавита (код ниже) (рис. [2.6]).



```
vpkozlov@fedora:~/lab12
[vpkozlov@fedora ~]$ cd ~/lab12
[vpkozlov@fedora lab12]$ vi lab12_3
[vpkozlov@fedora lab12]$ chmod +x lab12_3
[vpkozlov@fedora lab12]$ ./lab12_3
chddj
[vpkozlov@fedora lab12]$
```

Рис. 2.6: Создание и запуск файла_3

Код командного файла, генерирующего случайную последовательность букв латинского алфавита (рис. [2.7]).



```
vpkozlov@fedora:~/lab12 — /usr/libexec/vi lab12_3
echo $RANDOM | tr -d '0-9' | tr -d 'a-zA-Z'
```

Рис. 2.7: Код файла_3

3 Выводы

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

4 Ответы на 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 языков 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