

Лабораторная работа №13

Операционные системы

Гусева Светлана Алексеевна

Содержание

1 Цель работы	5
2 Задание	6
3 Выполнение лабораторной работы	7
4 Выводы	13
5 Контрольные вопросы	14

Список таблиц

Список иллюстраций

3.1 Результат выполнения задания 1 в двух терминалах.	8
3.2 Результат выполнения задания 1 в трех терминалах.	8
3.3 Скрипт задания 1.	9
3.4 Результат выполнения задания 2.	10
3.5 Результат выполнения задания 2.	10
3.6 Скрипт задания 3.	11
3.7 Результат выполнения задания 3.	12

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. 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.

Пока существует файл с названием `sem_file`, ожидается освобождение ресурса. Затем ресурс используется в течение времени t_2 (создается файл `sem_file`, происходит ожидание в течение t_2 секунд, после чего `sem_file` удаляется). Результат выполнения задания 1 в двух терминалах (рис. 3.1), результат выполнения задания 1 в трех терминалах (рис. 3.2), скрипт задания 1 (рис. 3.3).

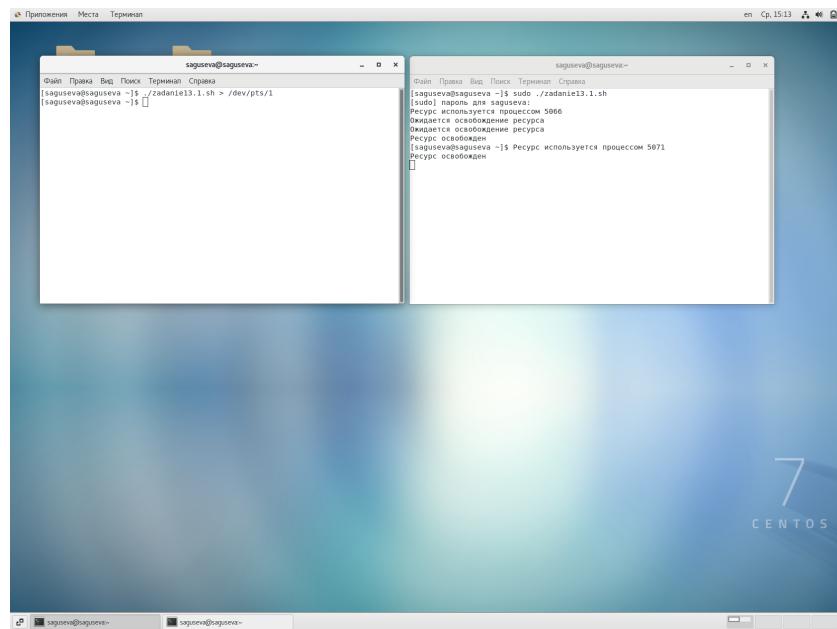


Рис. 3.1: Результат выполнения задания 1 в двух терминалах.

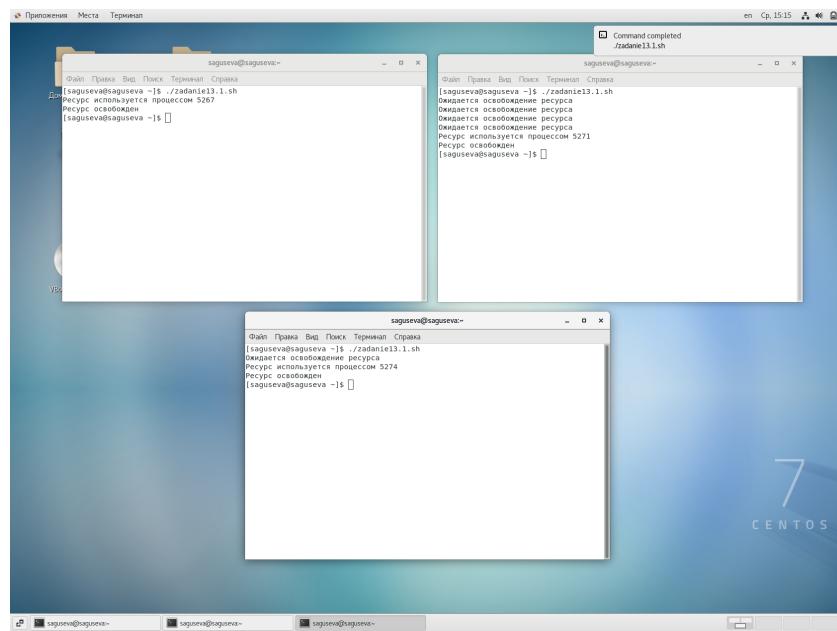


Рис. 3.2: Результат выполнения задания 1 в трех терминалах.

The screenshot shows a Linux desktop environment with a terminal window and an Emacs window. The terminal window, titled 'zadanie13.1.sh - emacs@zagueeva.localdomain', contains the following shell script:

```

#!/bin/bash
#> echo "Ресурс используется процессом $2"
#> sleep $2
#> rm "$sem_file"
#> echo "Ресурс освобожден"

```

The Emacs window, titled 'zadanie13.1.sh - emacs@zagueeva.localdomain', shows the same script with syntax highlighting. The desktop background is a blue gradient with the CentOS 7 logo.

Рис. 3.3: Скрипт задания 1.

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`.

Если задан параметр и существует архив с данным названием, то выводится справка. Если параметр не задан или архива не существует, выводится сообщение об ошибке (рис. 3.4, рис. 3.5).

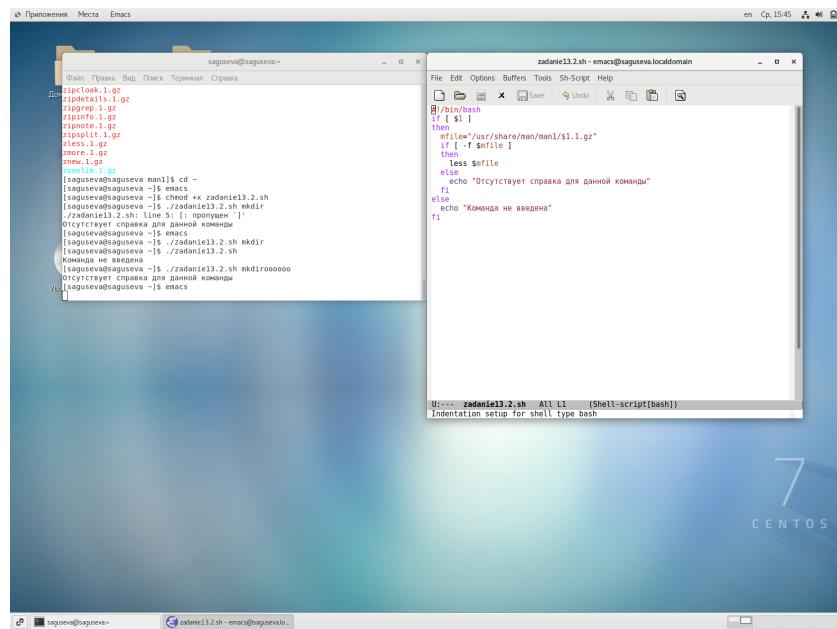


Рис. 3.4: Результат выполнения задания 2.

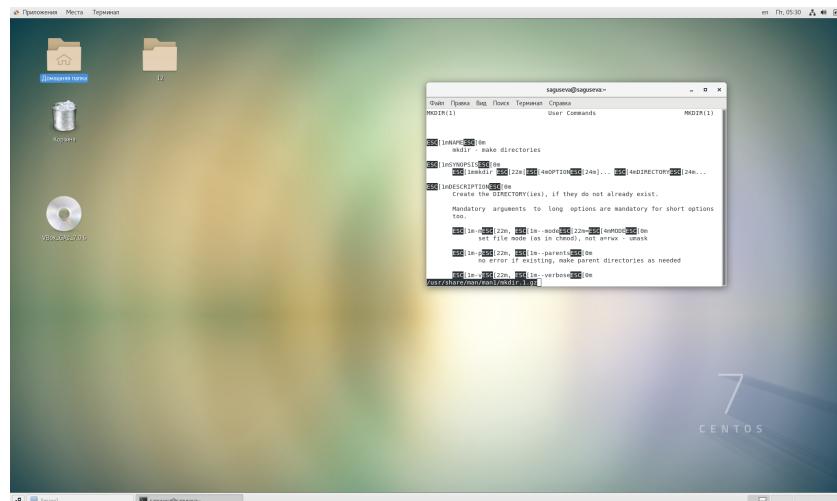


Рис. 3.5: Результат выполнения задания 2.

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.

Сначала с помощью \$RANDOM определяется число l, в моей программе оно меньше 50,

чтобы последовательность символов не была слишком длинной. Далее в цикле от 0 до 1 с помощью \$RANDOM определяется число num, принадлежащее интервалу от 65 до 122 (эти числа соответствуют латинским буквам и некоторым знакам, номера знаков от 91 до 96). Если числу num соответствует буква, то оно добавляется в последовательность, если соответствует символ, то это число не включается в последовательность. Программа (рис. 3.6) и результат выполнения (рис. 3.7).

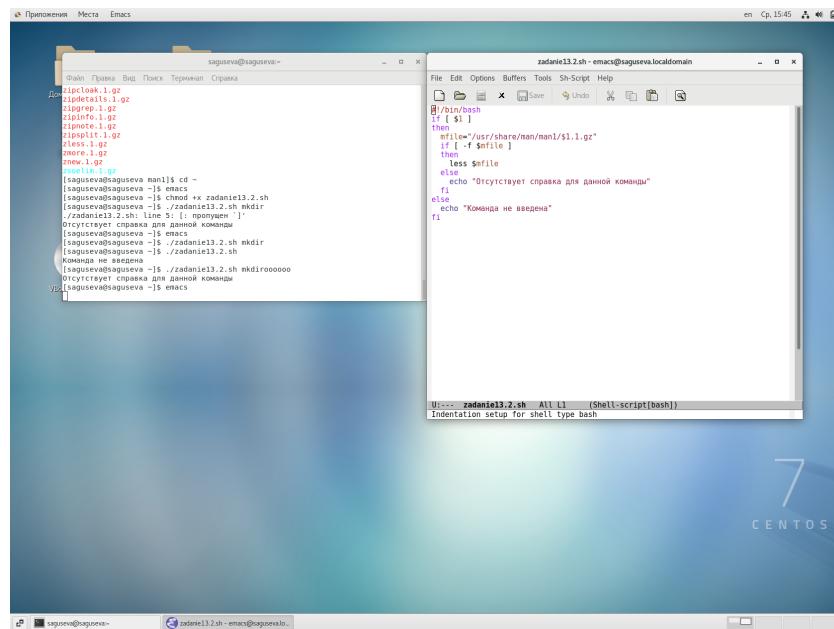


Рис. 3.6: Скрипт задания 3.

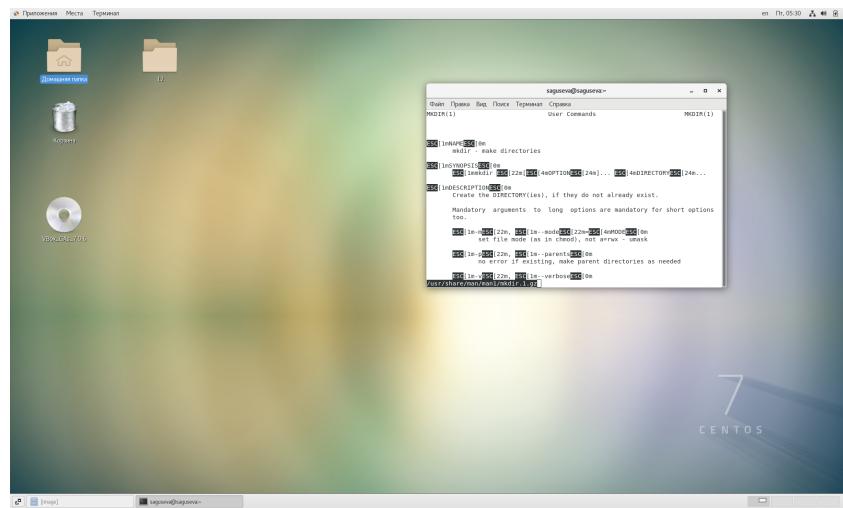


Рис. 3.7: Результат выполнения задания 3.

4. Выводы

В процессе выполнения лабораторной работы мной были изучены основы программирования в оболочке ОС UNIX. Я научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5. Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

```
while [$1 != "exit"]
```

Без ошибки: while ((\$1 != "exit")) или while [\$1 -ne "exit"]

2. Как объединить (конкатенация) несколько строк в одну?

```
s1="Hello,"  
s2=" World"  
s3="s1s2"
```

3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

seq — утилита, используемая в unix-системах для генерации последовательности. В самом простом варианте использования — seq N — выводит на печать все целые числа от 1 до N в последовательности. Для получения последовательности целых чисел вместо seq можно использовать цикл for.

4. Какой результат даст вычисление выражения \$((10/3))?

3

5. Укажите кратко основные отличия командной оболочки zsh от bash.

У Zsh есть поддержка с плавающей точкой, которой нет у Bash.

В Zsh поддерживаются структуры хеш-данных, которых нет в Bash.

Функции вызова в Bash лучше по сравнению с Zsh.

Внешний вид подсказки можно контролировать в Bash, тогда как Zsh настраивается.

Конфигурационными файлами являются .bashrc в интерактивных оболочках без регистрации и .profile или .bash_profile в оболочках входа в Bash. В Zsh оболочками, не входящими в систему, являются .zshrc, а оболочками для входа - .zprofile.

Массивы Zsh индексируются от 1 до длины, тогда как Bash индексируется от -1 до длины.

В Zsh, если шаблоны не совпадают ни с одним файлом, выдается ошибка. Находясь в Баше, он остается без изменений.

Правая часть конвейера запускается как родительская оболочка в Zsh, в то время как в Bash она запускается как подоболочка.

В Zsh функция zmv используется для массового переименования, тогда как в Bash мы должны использовать функцию расширения параметров.

Bash имеет хорошие возможности написания сценариев в одной строке, в то время как в Zsh мы не смогли найти то же самое.

По умолчанию выходные данные хранятся во временном файле в Zsh, а в Bash - нет.

Многие встроенные функции в Bash упрощают сложные программы, тогда как в Zsh встроенных функций для сложных программ меньше.

Zsh эффективно управляет своими файлами, в то время как Bash плохо умеет работать с файлами.

6. Проверьте, верен ли синтаксис данной конструкции

```
for ((a=1; a <= LIMIT; a++))
```

Верен. До цикла переменной LIMIT должно быть присвоено значение.

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

Проведу сравнение bash и python. Bash предназначен для Linux и macOS для автоматизации задач. Он подходит для простых и небольших скриптов. Python предназначен для разработки

веб-приложений и приложений. Он подходит для больших программ. Python проще в использовании, его синтаксис интуитивно понятнее. Программы, написанные на языке Python, можно запускать на любой платформе, скрипты на Bash можно запускать только на Linux'е. Bash был разработан для системных администраторов для выполнения автоматизированных задач. И наоборот, Python был создан для программистов для разработки веб-сайтов и приложений. Следовательно, оба языка программирования были созданы для разных целей.