Отчёт по лабораторной работе № 12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Нати Франшиску Бунда

Содержание

# 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

* Ознакомиться с теоретическим материалом.
* Выполнить упражнения.
* Ответить на контрольные вопросы.

# 3 Выполнение лабораторной работы

## 3.1 Написание программ

1. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, 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 (> /dev/tty#, где # — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. (рис. [[1](#fig:001)]), (рис. [[2](#fig:002)]), (рис. [[3](#fig:003)]), (рис. [[4](#fig:004)])

Figure 1: Скрипт 1

Figure 1: Скрипт 1



Figure 2: Скрипт 1

Figure 3: Скрипт 1

Figure 3: Скрипт 1

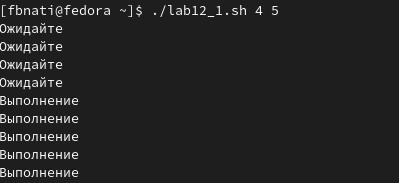


Figure 4: Скрипт 1

Доработали программу так, чтобы имелась возможность взаимодействия трёх и более процессов.(рис. [[5](#fig:005)]), (рис. [[6](#fig:006)])



Figure 5: Скрипт 1, доработка

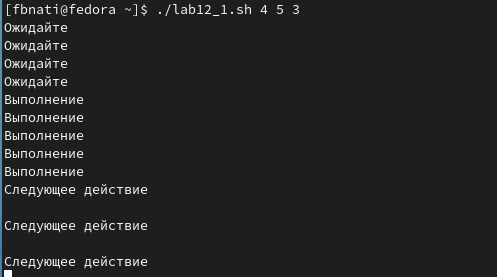


Figure 6: Скрипт 1, доработка

1. 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. Командный файл получает в виде аргумента командной строки название команды и в виде результата выдаёт справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1. (рис. [[7](#fig:007)]), (рис. [[8](#fig:008)]), (рис. [[9](#fig:009)]), (рис. [[10](#fig:010)]), (рис. [[11](#fig:011)]), (рис. [[12](#fig:012)]), (рис. [[13](#fig:013)])

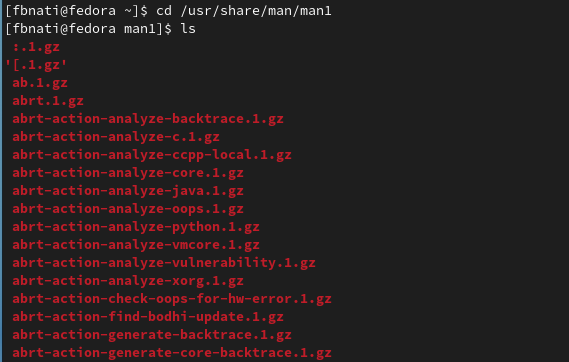


Figure 7: /usr/share/man/man1

Figure 8: Скрипт 2

Figure 8: Скрипт 2



Figure 9: Скрипт 2

Figure 10: Скрипт 2

Figure 10: Скрипт 2

Figure 11: Скрипт 2

Figure 12: Скрипт 2

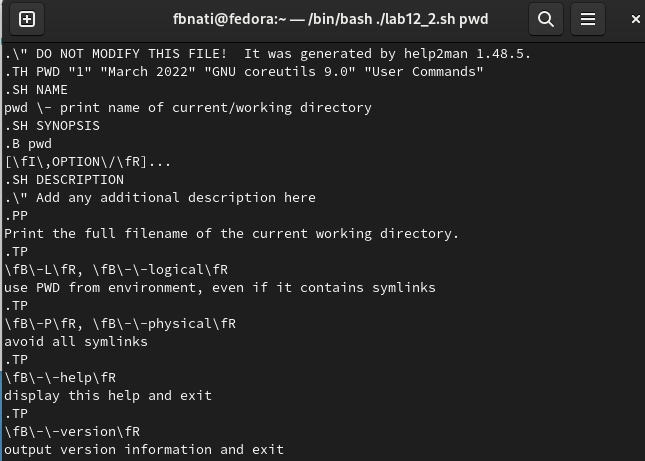


Figure 13: Скрипт 3

1. Используя встроенную переменную $RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита, учитывая, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. [[14](#fig:014)]), (рис. [[15](#fig:015)]), (рис. [[16](#fig:016)]), (рис. [[17](#fig:017)])

Figure 14: Скрипт 3

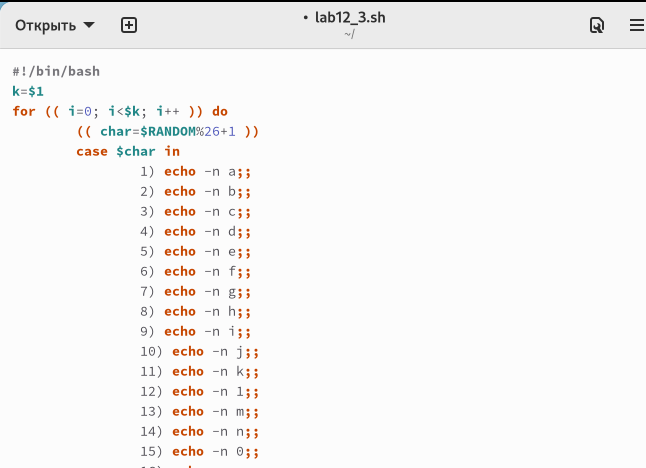


Figure 15: Скрипт 3

Figure 16: Скрипт 3

Figure 17: Скрипт 3

# 4 Выводы

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

# 5 Ответы на контрольные вопросы

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. Чтобы объединить несколько строк в одну, можно воспользоваться несколь- кими способами:

Первый: VAR1=”Hello, “VAR2=” World” VAR3=“𝑉 𝐴𝑅1VAR2” echo “VAR1” Результат: Hello, World

1. 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 выравнива- ния ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.
2. Результатом данного выражения $((10/3))будет 3, потому что это целочис- ленное деление без остатка.
3. Отличия командной оболочки zshот bash:

В zsh более быстрое автодополнение для cdс помощью Тab В zsh существует калькулятор zcalc, способный выполнять вычисления внут- ри 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 поддерживаетсязаменачастипути В zsh есть возможность отображать разделенный экран, такой же как разде- ленный экран vim

1. for((a=1; a<= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().
2. Преимущества скриптового языка bash:

Один из самых распространенных и ставится по умолчаниюв 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 быстроте выполнения этого скрипта Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий