Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Крутова Екатерина Дмитриевна

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

1. Написать 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). 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

## 3.1 Задание 1 (рис. 1-2)

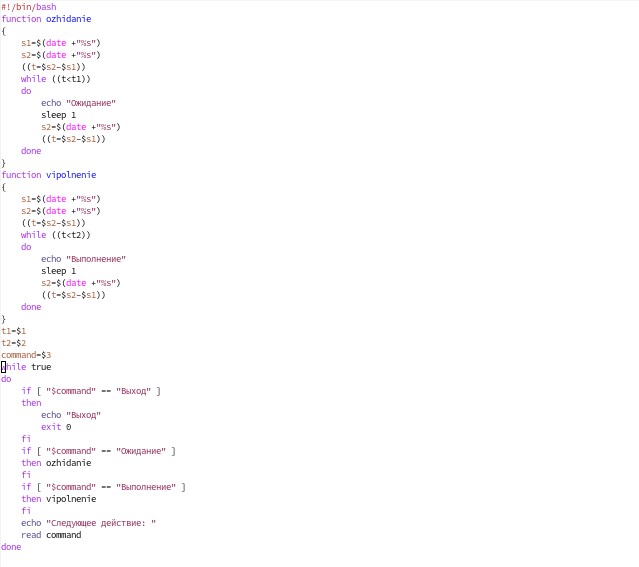


Рис. 1: Текст командного файла к заданию 1

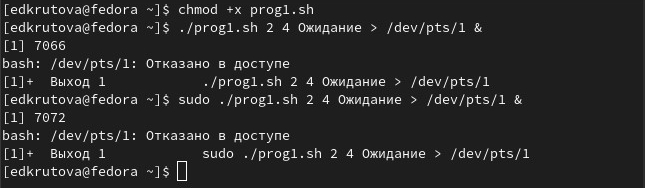


Рис. 2: Создание исполняемого файла и проверка

## 3.2 Задание 2 (рис. 3-5)

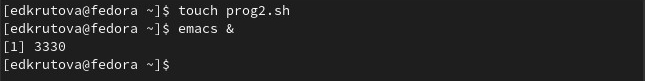


Рис. 3: Создание файла и вызов редактора

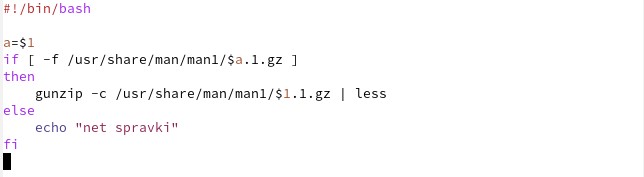


Рис. 4: Текст командного файла к заданию 2

Рис. 5: Создание исполняемого файла и проверка

Рис. 5: Создание исполняемого файла и проверка

## 3.3 Задание 3 (рис. 6-7)

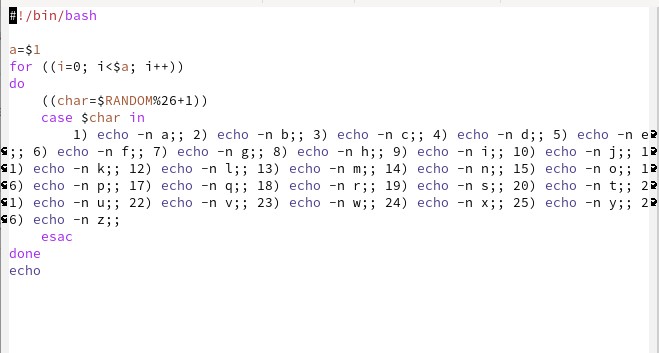


Рис. 6: Текст командного файла к заданию 3

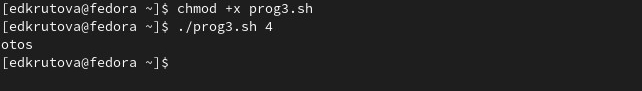


Рис. 7: Создание исполняемого файла и проверка

# 4 Выводы

Я изучила основы программирования в оболочке ОС UNIX. Научиться 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”]

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” ]

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

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:

• Первый: VAR1=“Hello,” VAR2=” World” VAR3=“VAR2” echo “$VAR3” Результат: Hello, World

• Второй: VAR1=“Hello,” VAR1+=” World” echo “$VAR1” Результат: Hello, World

1. 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.

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

Результатом данного выражения $((10/3)) будет 3, потому что это целочисленное деление без остатка.

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

• В zsh более быстрое автодополнение для cd с помощью Тab

• 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 основенеполных данных

• В zsh поддерживается замена части пути

• В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim

1. Проверьте, верен ли синтаксис данной конструкции

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

синтаксис данной конструкции верен

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

• 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 быстроте выполнения этого скрипта

• Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий