Лабораторная работа №13

Операционные системы

Тойчубекова Асель Нурлановна

Содержание

# 1 Цель работы

Целью данной лабораторной рабораторной работы является изучение основ программирования в оболочке OC Unix. Также 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, 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;
* С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
* оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
* BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

POSIX-совместимые оболочки разработаны на базе оболочки Корна.

# 4 Выполнение лабораторной работы

Для начала создаю файл в котором буду писать саму програму и открываю его в редакторе gedit (рис. 1).

Создание файла

Рис. 1: Создание файла

Редактирую файл, записывая командный файл, который реализует упрощенный механизм семофонов. Для этого я сперва создаю переменную в которой будет храниться адрес файла блокирования, а также переменную, которому присваивается значение первого аргумента. Дальше открыв файл блокирования с помощью команды exec и приваиваю ему свободный файловый дискриптор. Далее запускаю цикл, который будет работать пока файл блокирования существует и проверяю можно ли заблокировать фaйл c помощью команды flock -n, если да, то вывожу сообщение, что файл заблокирован, затем жду столько времени, сколько было указано в аргумете и разблокировываю файл с помощью команды flock -u, выведя об этом сообщение.Если же файл изначально был заблокирован, то я вывожу сообщения, что файл заблокирован и жду пока не истечет время, указанное в аргументе. (рис. 2).

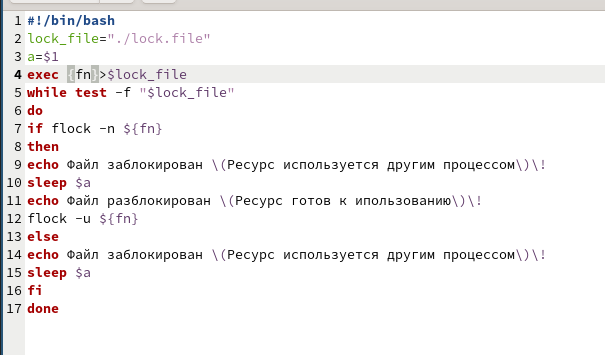


Рис. 2: Редактирование файла

Даю права на выполнение и запускаю программу, мы видим, что все работает корректно. (рис. 3).

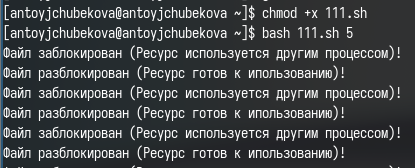


Рис. 3: Запуск программы

Создаю файл для написания программы второго задания. (рис. 4).

Создание файла

Рис. 4: Создание файла

Далее перехожу по ссылке /usr/share/man/man1 и вижу архивы текстовых файлов, содержащих справки о командах. (рис. 5).

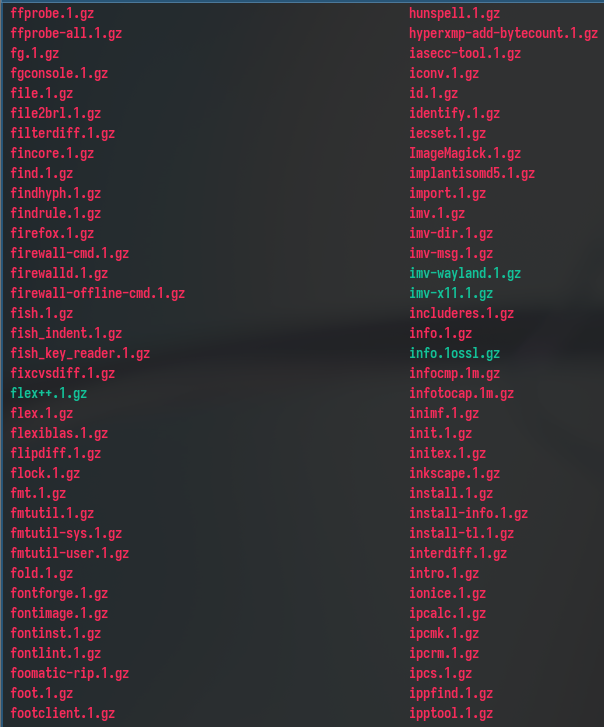


Рис. 5: Справки о командах

Открываю созданный файл в редакторе и редактирую его, записывая командный файл, который реализует команду man, перейдя по адресу /usr/share/man/man1 и используя архивы данных о командах выполняет эту команду. Для этого сперва завожу переменную, которой присваиваю значения аргуметна, затем проверяю есть ли подходящий файл в man1, если есть с помощью команды less вывожу ее, а если нет вывожу соответствующее сообщение. (рис. 6).

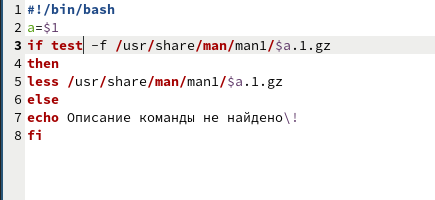


Рис. 6: Редактирование файла

Даю права на выполнение и запускаю программу, мы видим, что все работает правильно и программа выдает нам сообщение о команде. (рис. 8).

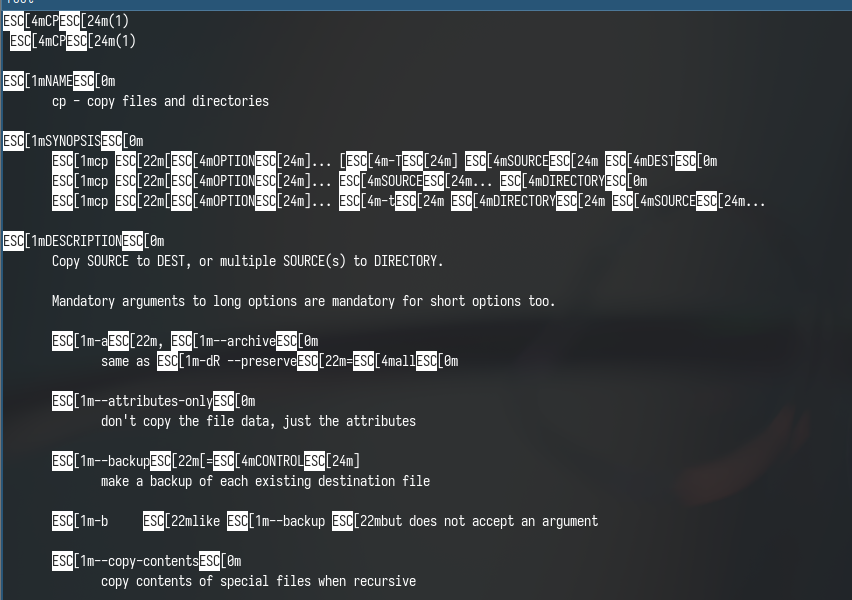


Рис. 7: Запуск программы

Создаю файл для написание программы для третьего задания и открываю его в редакторе. (рис. 8).

Создание файла

Рис. 8: Создание файла

Редактирую файл, записывая командный файл, который используя встроенную переменную $RANDOM, генерирует случайную последовательность букв латинского алфавита. Для этого я создаю переменную и присваиваю ей значения аргумента, далее запускаю цикл, который будет продолжаться пока i не станет равным аргументу, запускаю RANDOM и в соответствии с получившимся числом с помощью командыcase перебираю по цифрам все буквы, если совпадает вывожу на экран, в конце перехожу на новую строку. (рис. 9).

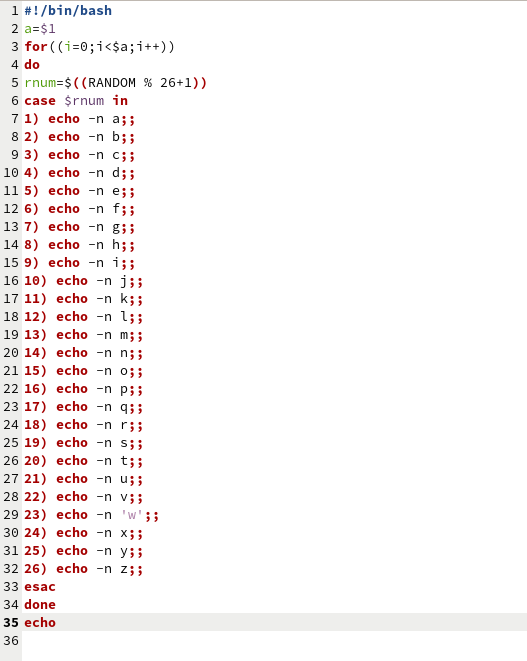


Рис. 9: Редактирование файла

Даю права на выполнение и запускаю программу, мы видим, что программа правильно работает и выводит латинские букввы. (рис. 10).

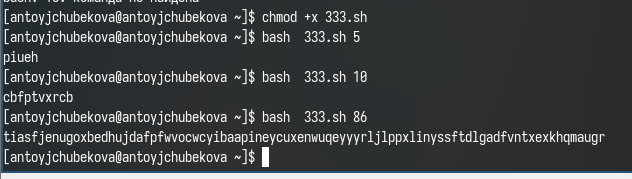


Рис. 10: Запуск программы

# 5 Ответы на вопросы

1. 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.
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: VAR1=“Hello,” VAR2=” World” VAR3=“VAR2” echo “VAR1” Результат: Hello, World.
3. 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. Результатом данного выражения $((10/3)) будет 3, потому что это целочисленное деление без остатка.
5. 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.
6. for ((a=1; a <= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().
7. Преимущества и недостатки 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 быстроте выполнения этого скрипта;
* Скрипты, написанные на bash, нельзя запустить на других операционныхсистемах без дополнительных действий;

# 6 Выводы

В ходе лабораторной работы №14 я изучила основы программирования в оболочке OC Unix. Также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.