Операционные системы

Лабораторная работа №12

Матюшкин Денис Владимирович (НПИбд-02-21)

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 Ход работы

1. Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустим 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. Доработаем программу так, чтобы имелась возможность взаимодействия трёх и более процессов (рис. 1). Проверим написанный командный файл (рис. 2).

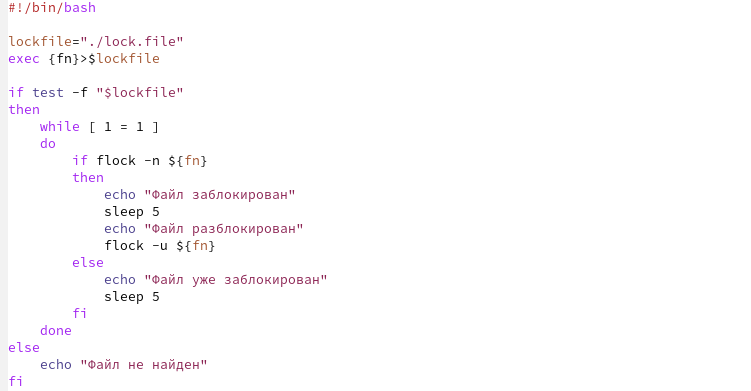


Рис. 1: Командный файл

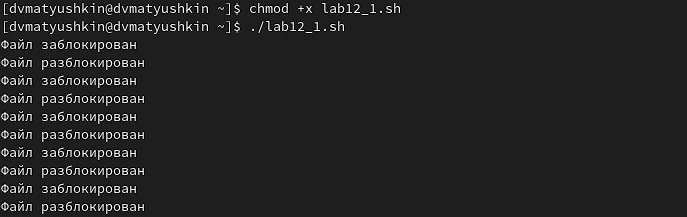


Рис. 2: Проверка файла

1. Реализуем команду man с помощью командного файла. Изучим 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 получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1 (рис. 3). Проверим написанный командный файл (рис. 4 и рис. 5).

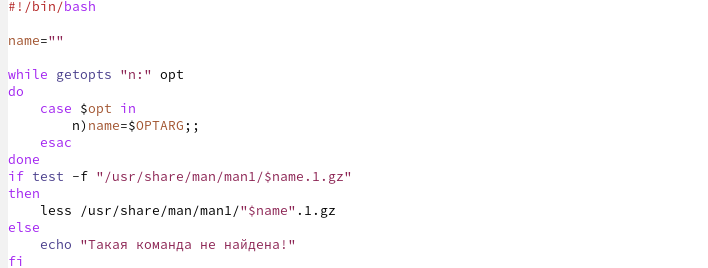


Рис. 3: Командный файл

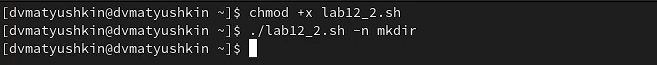


Рис. 4: Проверка файла

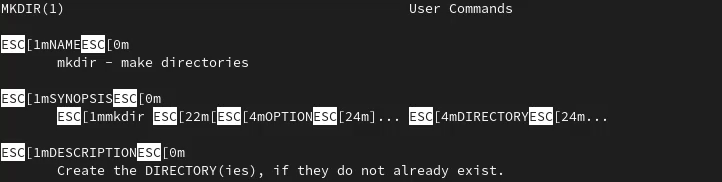


Рис. 5: Проверка файла

1. Используя встроенную переменную $RANDOM, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтем, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767 (рис. 6). Проверим написанный командный файл (рис. 7).



Рис. 6: Командный файл

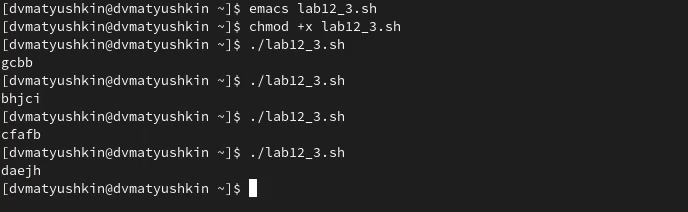


Рис. 7: Проверка файла

# 3 Контрольные вопросы

**1. Найдите синтаксическую ошибку в следующей строке: while [$1 != “exit”]**  
Ответ: $1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

**2. Как объединить (конкатенация) несколько строк в одну?**  
cat file.txt | xargs | sed …

**3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?**  
seq - выдает последовательность чисел.  
Реализовать ее функционал можно командой:

for n in {1..5}   
do <КОМАНДА>   
done

**4. Какой результат даст вычисление выражения $((10/3))?**  
Ответ: 3

**5. Укажите кратко основные отличия командной оболочки zsh от bash.**  
Zsh очень сильно упрощает работу. Но существуют различия. Например, в zsh после for обязательно вставлять пробел, нумерация массивов в zsh начинается с 1 (что не особо удобно на самом деле).  
Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендуется Bash. Если скрипты вам не нужны - Zsh (более простая работа с файлами, например).

**6. Проверьте, верен ли синтаксис данной конструкции for ((a=1; a <= LIMIT; a++))**  
Ответ: Верен.

**7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?**  
Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличи от обычного языка программирования). Но относительно обычных языков программирования bash очень сжат. Тот же Си имеет гораздо более широкие возможности для разработчика.

# 4 Вывод

* В ходе этой лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.