Лабораторная работа №14

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Мухин Тимофей Владимирович

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 Выполнение лабораторной работы

1. Напишем командный файл, реализующий упрощённый механизм семафоров. Ко- мандный 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.

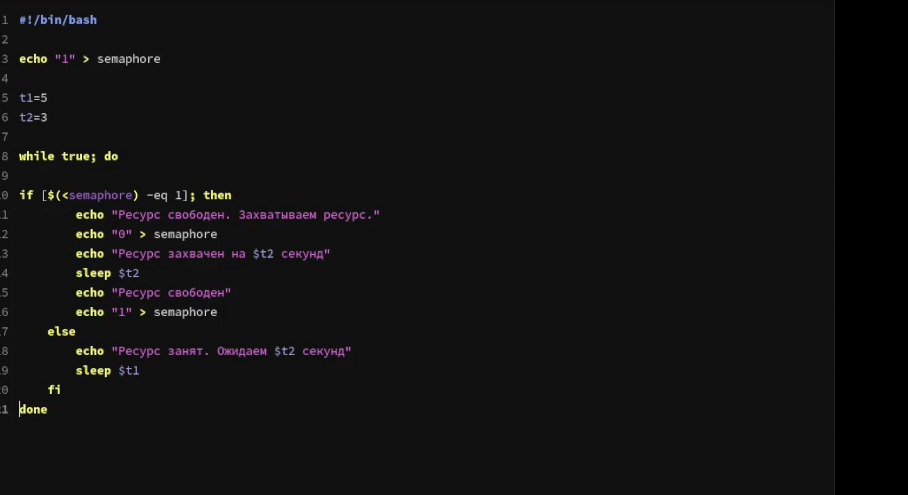


Рис. 1: Скрипт lab14-1

1. Запустим

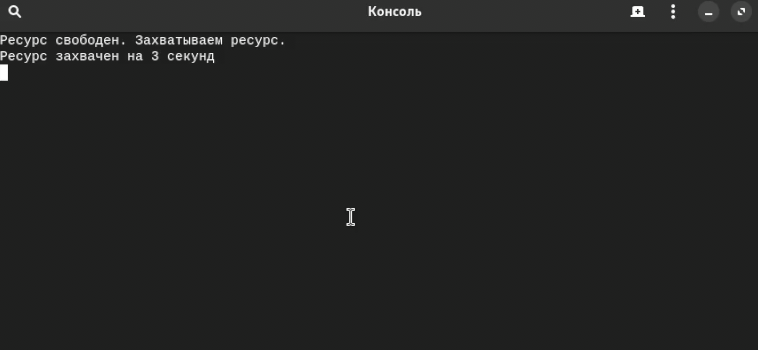


Рис. 2: Выполнение

1. Реализовать команду man с помощью командного файла. Изучите содержимое ката- лога /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

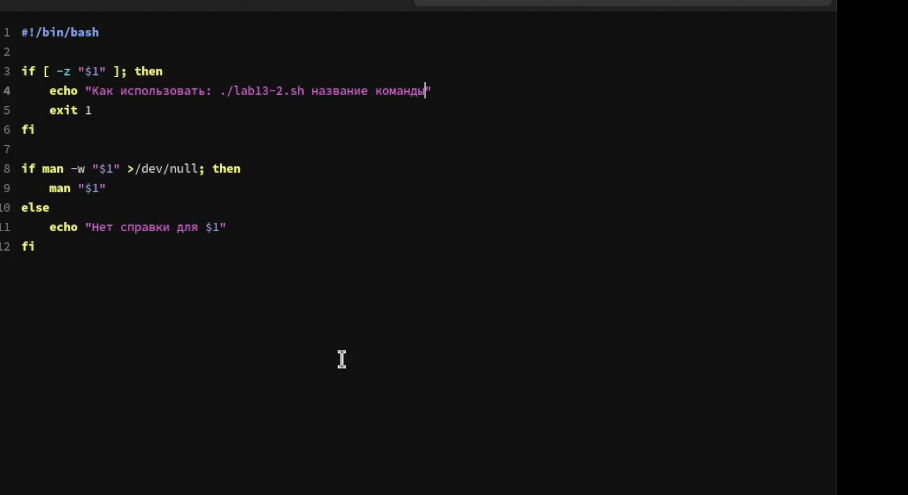


Рис. 3: Скрипт lab14-2

1. Запустим

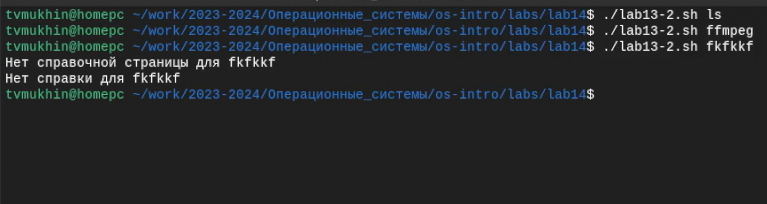


Рис. 4: Выполнение

1. Используя встроенную переменную $RANDOM, напишите командный файл, генерирую- щий случайную последовательность букв латинского алфавита. Учтите, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767



Рис. 5: Скрипт lab14-3

1. Запустим

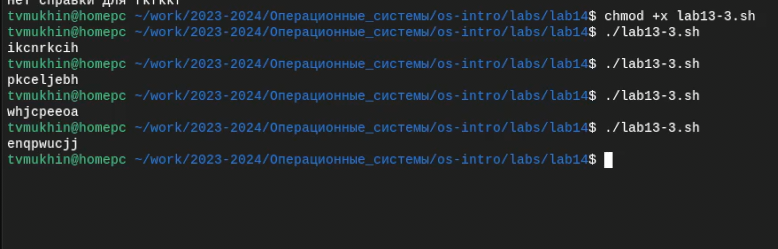


Рис. 6: Выполнение

# 3 Контрольные вопросы

1. Синтаксическая ошибка в строке while [$1 != "exit"]: пробелы обязательны перед и после квадратных скобок. Правильный вариант: while [ "$1" != "exit" ].
2. Для объединения (конкатенации) строк в bash можно использовать оператор +=. Например:

str1="Hello"  
str2="World"  
result="$str1 $str2"  
echo $result # Результат: Hello World

1. Утилита seq используется для генерации последовательностей чисел. Ее функционал можно реализовать через циклы или операторы for в bash. Например:

# С использованием цикла for  
for i in {1..5}; do  
 echo $i  
done  
  
# С использованием оператора for  
for ((i=1; i<=5; i++)); do  
 echo $i  
done

1. Выражение $((10/3)) даст результат целочисленного деления чисел 10 на 3, то есть 3.
2. Основные отличия командной оболочки zsh от bash:
   * zsh имеет более продвинутый автодополнитель и расширенную возможность конфигурации.
   * zsh поддерживает расширенный синтаксис, включая различные улучшения по сравнению с bash.
   * zsh предоставляет более продуманный механизм работы с командами и плагины.
3. Правильный синтаксис конструкции for в bash:

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

1. Сравнение bash с другими языками программирования:
   * Преимущества bash:
     + Простота и удобство для написания скриптов командной оболочки.
     + Встроенная обработка командной строки, файлов, директорий.
     + Широкие возможности работы с системными службами.
   * Недостатки bash:
     + Ограниченные возможности по сравнению с полноценными языками программирования.
     + Низкая производительность и эффективность выполнения сложных операций.
     + Ограниченная масштабируемость для крупных проектов.

# 4 Выводы

В ходе выполнения работы я изучил основы программирования в ос Linux и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.