Отчёт по лабораторной работе

Лабораторная работа №12

Хватов Максим Григорьевич

Содержание

Список иллюстраций

Список таблиц

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 Теоретическое введение

Вся необходимая теория была описана в прошлых лабораторных работах, она касается циклов и условий.

4 Выполнение лабораторной работы

1. Перва якоманда запускает скрипт в фоновом режиме с задержкой 5 секунд и временем использования 10 секунд. Вторая команда запускает скрипт в привилегированном режиме, перенаправляя вывод в терминал 2. После перенкаправления, в терминале 2 будет сообщение о том, что ресурс успешно использован. Комментарии к коду в листинге.

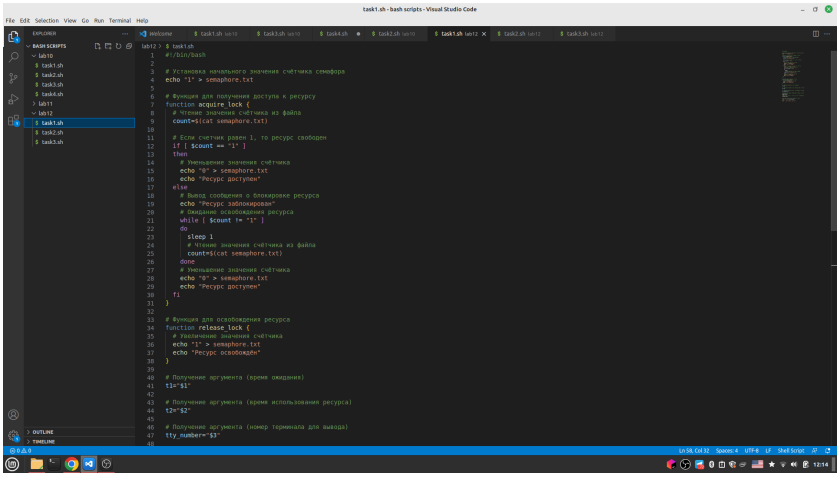

Рис. 4.1: Задание 1

2. Скрипт проверяет наличие архива справки для команды, указанной в аргументе командной строки. Если архив существует, то он распаковывается с помощью команды zcat и выводится с помощью less. Если архив не найден, то выводится сообщение об отсутствии справки.

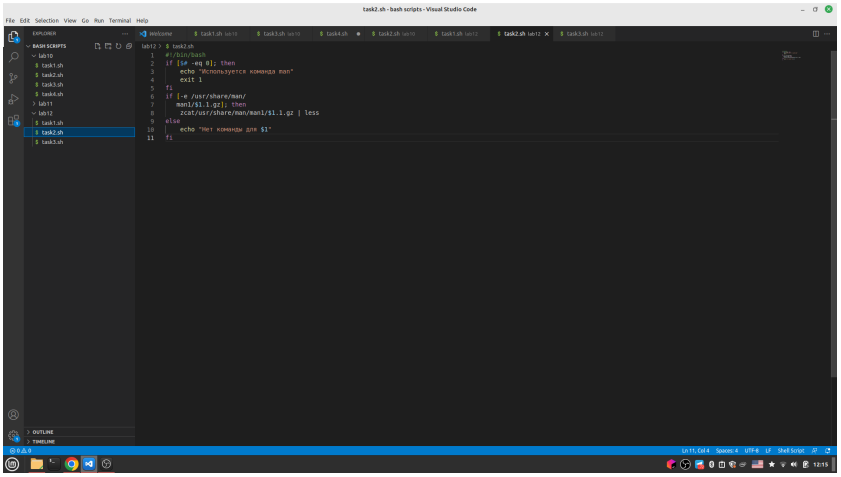

Рис. 4.2: Задание 2

3. В этом скрипте мы используем цикл for, чтобы сгенерировать последовательность из 10 букв латинского алфавита. Для каждого шага мы генерируем случайное число от 0 до 25. Затем мы преобразуем это число в соответсвующий символ латинского алфавита. И затем просто выводим этот символ. Код в листинге

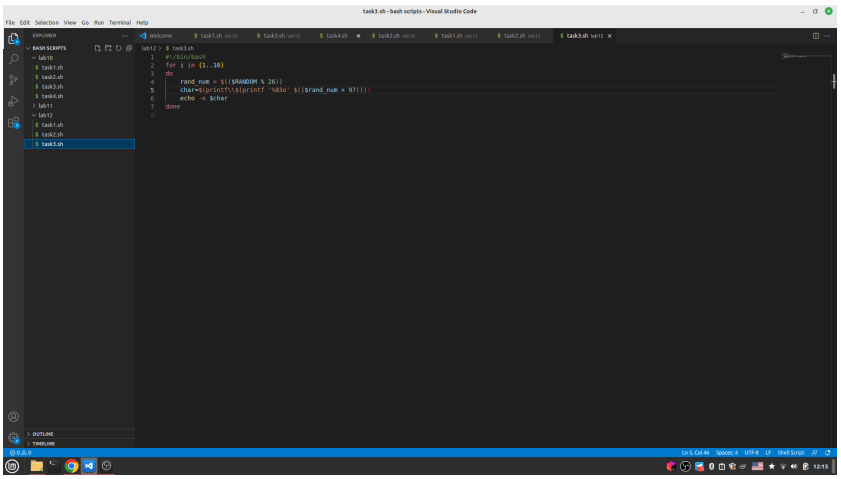

Рис. 4.3: Задание 3

5 Выводы

Я научился писать более сложные конструкции с помощью циклов и условынх операторов в Linux.

6 Ответы на контрольные вопросы

1. Синтаксическая ошибка в этой строке кода заключается в том, что в квадратных скобках должен быть используется знак равенства == вместо неравенства !=. Кроме того, скобки являются частью команды test, так что их необходимо оставить внутри обратных кавычек или двойных круглых скобок.
2. Вы можете использовать оператор “»”, чтобы добавить содержимое одной строки к другой строке. Например, вот как объединить строки “Hello” и “World!” в одну строку:

```
text="Hello "
text+="World!"
echo $text # выводит "Hello World!"
```

3. Использование синтаксиса фигурных скобок:

```
$ echo {1..10}
1 2 3 4 5 6 7 8 9 10
```

Использование цикла `for`:

```
$ for i in {1..10}; do echo $i; done
1
2
3
4
```

```
5
6
7
8
9
10
```

По сравнению с утилитой seq, эти способы могут быть более гибкими и позволяют создавать последовательности с более сложной логикой, например, с заданным шагом или с использованием других условий. Однако, утилита seq имеет простой и удобный синтаксис для создания простых последовательностей чисел.

4. 3,3333333333
5. Автодополнение в zsh работает более интеллектуально, позволяя просматривать и выбирать варианты рекурсивно.

В zsh есть более мощная поддержка плагинов и дополнений, таких как oh-my-zsh, что облегчает настройку и добавление функциональности.

В zsh есть более мощные возможности для работы с массивами, хэш-таблицами и переменными окружения.

В zsh есть встроенная поддержка расширенных шаблонов и условный оператор, позволяющий выполнять более сложные операции с файлами и директориями.

Значительное преимущество zsh заключается в более типизированном и безопасном программировании (например, строгая проверка типов и зарезервированные слова).

6. Верен
7. Преимущества языка bash:

Bash легче и быстрее в освоении, чем полноценные языки программирования, такие как Python или Perl.

Bash может использоваться для быстрого выполнения задач командной строки, таких как системное администрирование, обработка текста и сканирование файловой системы и т.д.

Bash имеет интегрированные сценарии управления файлами, поддержку командной строки и утилиты для редактирования текста.

Недостатки языка bash:

Bash не является полноправным языком программирования и может иметь ограниченный функционал для более сложных задач.

Синтаксис bash могут привести к сложным ошибкам, если быть неосторожным при написании скриптов.

Bash чувствителен к различным окончаниям строк, что может привести к сбоев в работе скриптов, если строковые окончания не будут корректно настроены.

Список литературы