

Архитектура компьютеров и операционные системы | Операционные системы

**Лабораторная работа № 12. Программирование в командном
процессоре ОС UNIX. Расширенное программирование**

Мугари Абдеррахим - НКАбд-03-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы :	6
2.1	Контрольные вопросы:	10
2.2	выводы по результатам выполнения заданий:	11
3	Выводы, согласованные с целью работы:	12

Список иллюстраций

2.1	Написание первой программы	6
2.2	выполнение первой программы	7
2.3	Написание второй программы	8
2.4	выполнение второй программы	9
2.5	Написание третьей программы	9
2.6	выполнение третьей программы	10

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 Выполнение лабораторной работы :

- Здесь я написал файл bash, реализующий упрощенный механизм семафора. Командный файл должен некоторое время $t1$ ждать освобождения ресурса, выдавая сообщение об этом, и после ожидания его освобождения использовать его в течение некоторого времени $t2 < t1$, также предоставляя информацию о том, что ресурс используется соответствующим командным файлом (процессом) (рис. 2.1)

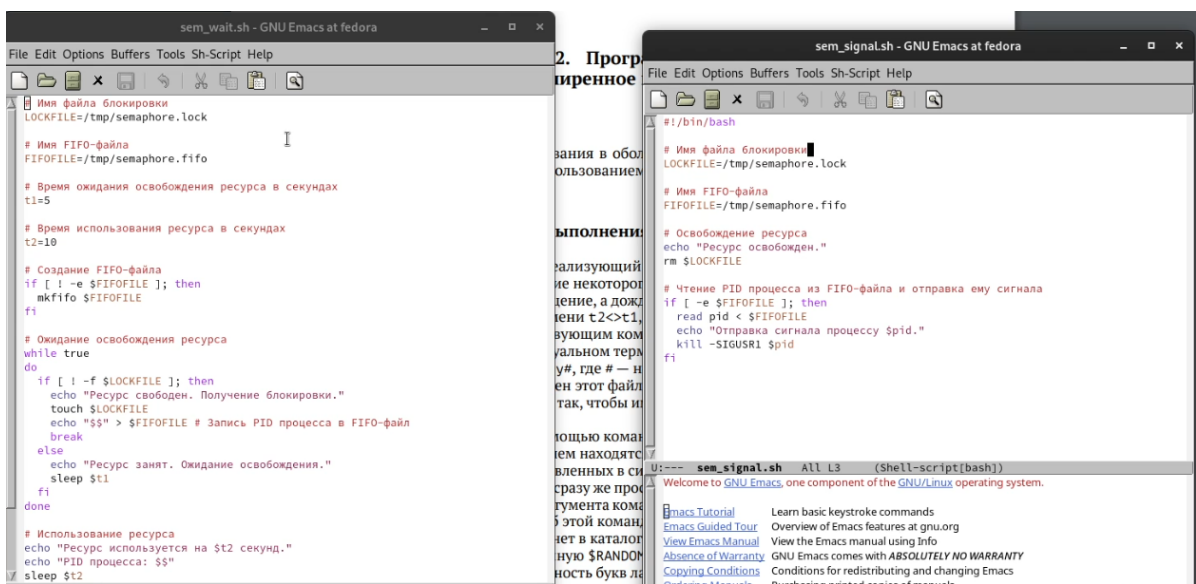


Рис. 2.1: Написание первой программы

- Здесь я выполнил код первой программы, и она заработала так, как и должна была работать (рис. 2.2)

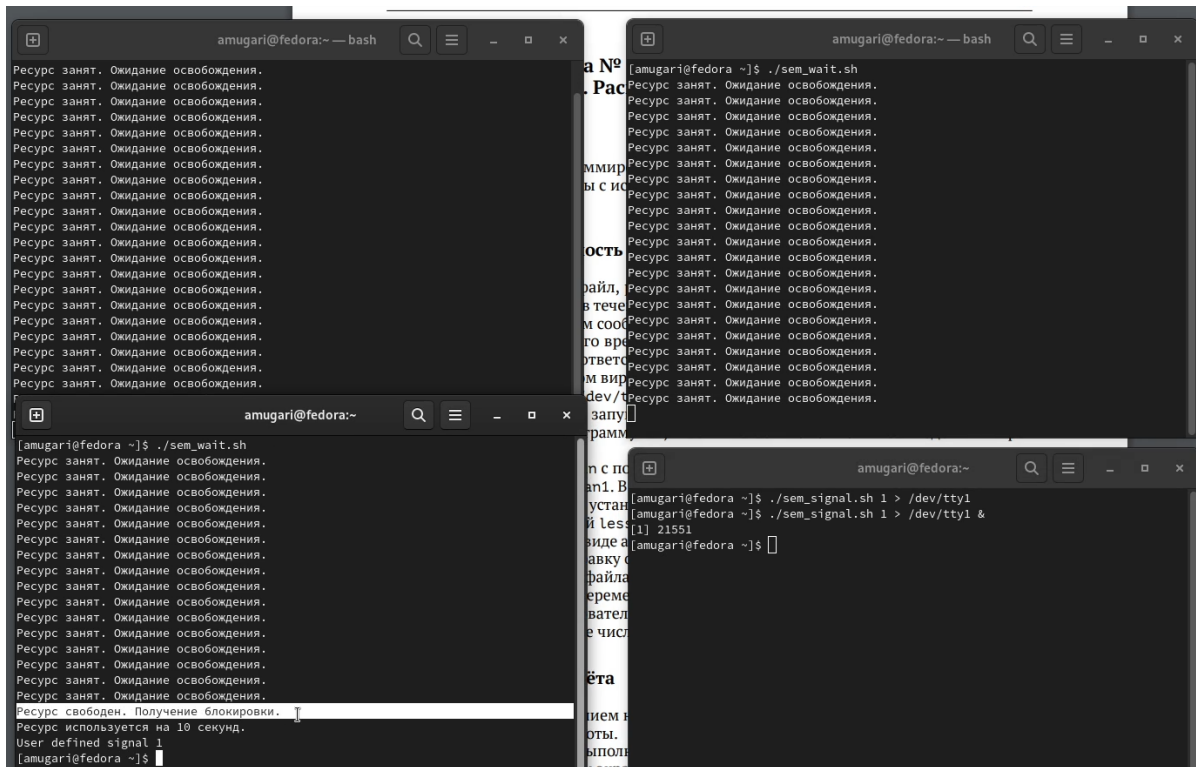


Рис. 2.2: выполнение первой программы

- После этого я написал скрипт bash, который использует содержимое `man log` `/usr/share/man/man1`. Он содержит архивы текстовых файлов, содержащих справку по большинству программ и команд, установленных в системе, а затем отображает справку по выбранной команде (рис. 2.3)

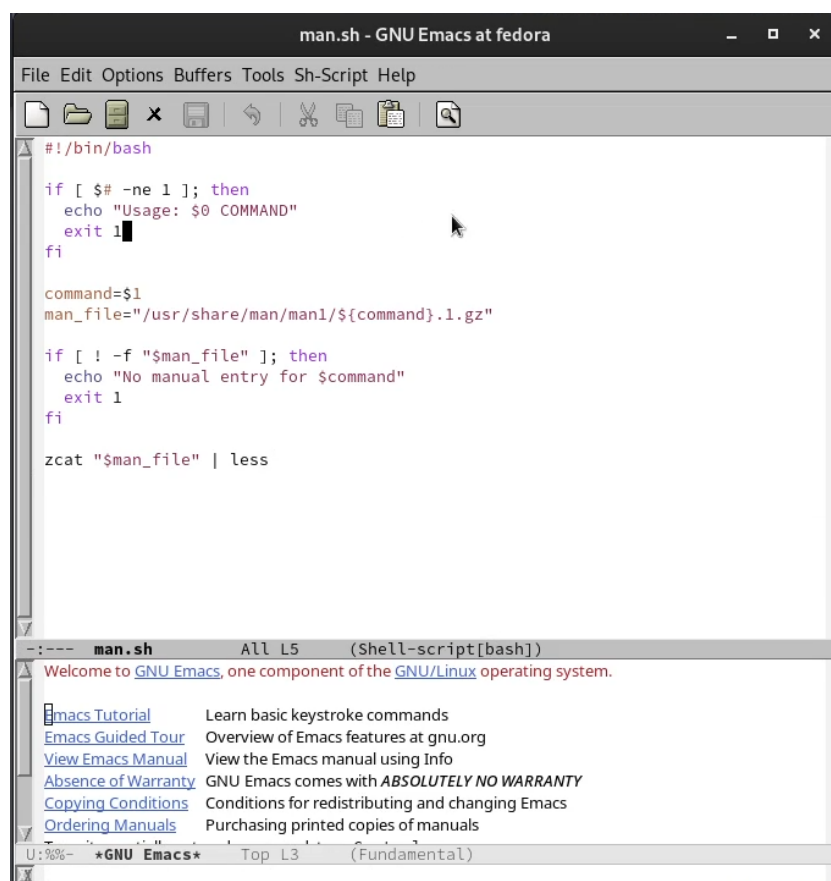
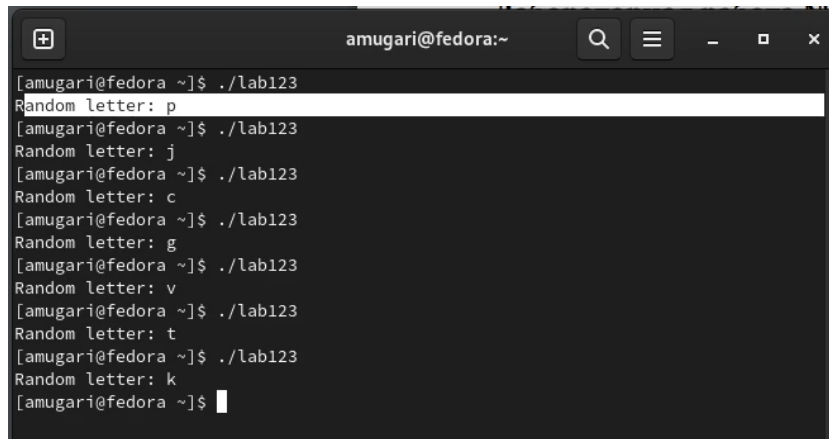


Рис. 2.3: Написание второй программы

- после этого я выполнил код скрипта, чтобы проверить справку команды sr, и он показал мне справку этой команды (рис. 2.4)



```
amugari@fedora:~  
[amugari@fedora ~]$ ./lab123  
Random letter: p  
[amugari@fedora ~]$ ./lab123  
Random letter: j  
[amugari@fedora ~]$ ./lab123  
Random letter: c  
[amugari@fedora ~]$ ./lab123  
Random letter: g  
[amugari@fedora ~]$ ./lab123  
Random letter: v  
[amugari@fedora ~]$ ./lab123  
Random letter: t  
[amugari@fedora ~]$ ./lab123  
Random letter: k  
[amugari@fedora ~]$
```

Рис. 2.6: выполнение третьей программы

2.1 Контрольные вопросы:

1. Синтаксическая ошибка в строке: необходимо заключить \$1 в двойные кавычки, чтобы избежать проблем с пробелами в аргументах. Также необходимо добавить скобки вокруг условия: `while ["$1" != "exit"]`
2. Для объединения нескольких строк в одну можно использовать оператор конкатенации `.` (точка) или просто перенос строки с использованием обратного слеша `\`.
3. Утилита `seq` предназначена для генерации последовательностей чисел. Она принимает три аргумента: начальное значение, конечное значение и шаг. Можно реализовать аналогичный функционал с помощью цикла `for` или `while` в `bash`.
4. Результат вычисления выражения `$((10/3))` будет равен 3. При делении целых чисел результат округляется в меньшую сторону.
5. `Bash` и `zsh` - это обе командные оболочки `Unix`. Основные отличия между ними заключаются в том, что `zsh` предоставляет больше возможностей для

автодополнения, расширенную подсветку синтаксиса и более продвинутые возможности встроенного языка программирования.

6. Синтаксис данной конструкции верен. Это цикл `for`, который использует арифметическое выражение для задания начального значения, условия продолжения цикла и шага.
7. Bash - это язык сценариев оболочки Unix. Он имеет синтаксис, похожий на язык программирования C, и предназначен для автоматизации задач командной строки. Он удобен для написания скриптов для автоматизации рутинных задач и не требует специальных знаний программирования. Преимущества `bash` включают простоту использования, мощные возможности текстовой обработки и доступность на большинстве Unix-подобных систем. Недостатки `bash` включают ограниченные возможности встроенного языка программирования и ограниченную поддержку многопоточности. В сравнении с другими языками программирования, такими как Python или Ruby, `bash` не так мощен и не предоставляет таких возможностей для разработки крупномасштабных приложений.

2.2 выводы по результатам выполнения заданий:

- В ходе этой лабораторной работы у меня была возможность научиться программировать в операционной системе UNIX и приобрести практические навыки написания сложного кода с использованием логических управляющих структур и циклов.

3 Выводы, согласованные с целью работы:

- Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.