Лабораторная работа № 12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Шулуужук Айраана Вячеславовна НПИбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Написать 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.

2. 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.

3. 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.

3 Теоретическое введение

3.1 Командные процессоры (оболочки)

Командный процессор (командная оболочка, интерпретатор команд shell) — это про- грамма, 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; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления програм- мой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей сов- мещает свойства оболочек C 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 опера- ционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash.

4 Выполнение лабораторной работы

Напишем командный файл, реализующий упрощенный механизм семафоров (рис. 4.1)

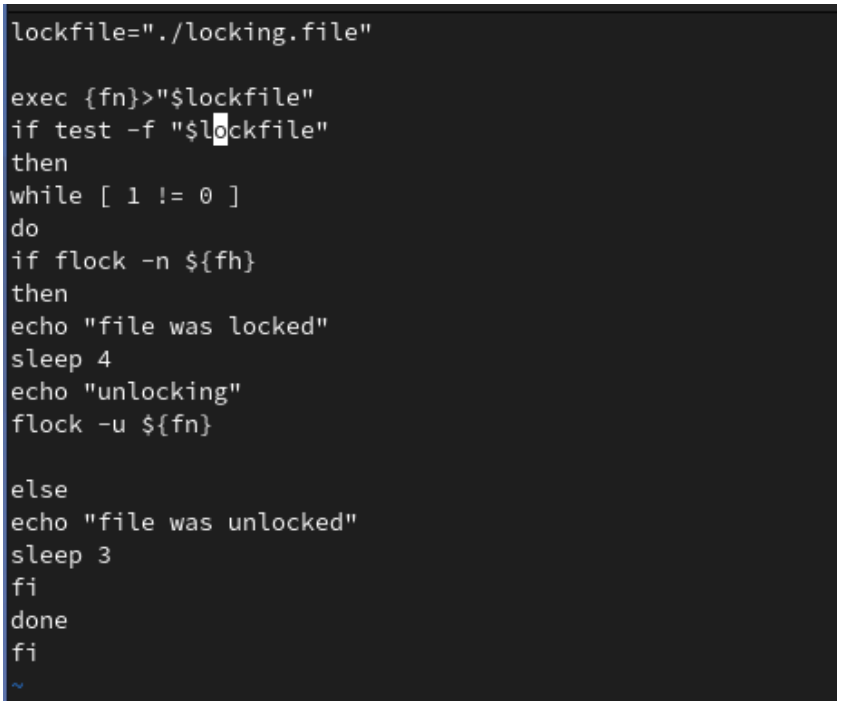

Рис. 4.1: скрипт 1

Скомпилируем данный файл и проверим его работу (рис. 4.2)

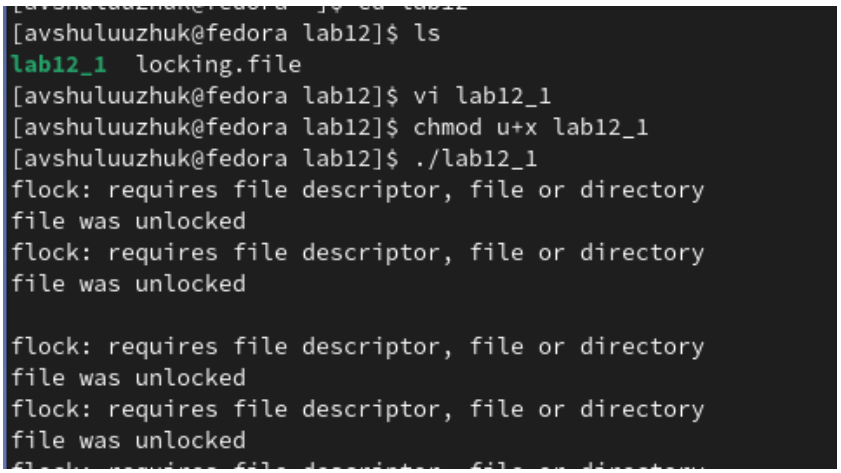

Рис. 4.2: результат работы командного файла 1

Изучим содержимое каталога /usr/share/man/man1, в котором находятся фрхивы текстовых файлов, содержащих справку о командах. (рис. 4.3)

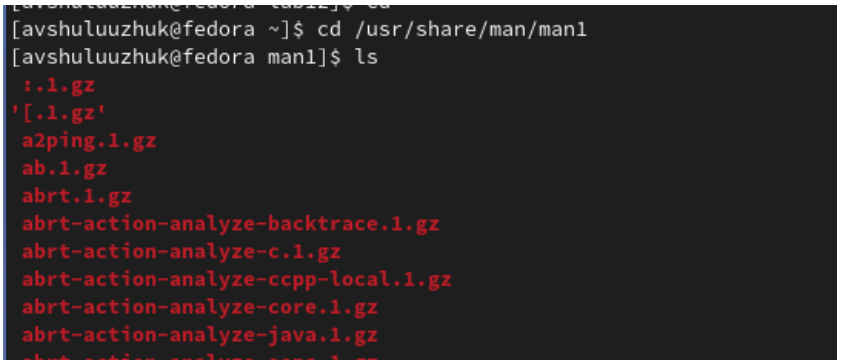

Рис. 4.3: содержиние каталога man1

Далее реализуем команду man с помощью командного файла (рис. 4.4)

```
command=""

while getopts :c: opt
do
case $opt in
c)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "no such a command"
fi
```

Рис. 4.4: скрипт 2

Проверим работу командного файла, используя нужную опцию и команду в качестве аргумента (рис. 4.5) (рис. 4.6)

```
[avshuluuzhuk@fedora lab12]$ vi lab12_2
[avshuluuzhuk@fedora lab12]$ chmod u+x lab12_2
[avshuluuzhuk@fedora lab12]$ ./lab12_2 -c ls
```

Рис. 4.5: результат запуска скрипта 2

```
TOUCH(1)                          User Commands                          TOUCH(1)

ESC[1mNAMEESC[0m
       touch - change file timestamps

ESC[1mSYNOPSISESC[0m
       ESC[1mtouch ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mFILEESC[24m...

ESC[1mDESCRIPTIONESC[0m
       Update the access and modification times of each FILE to the current time.
```

Рис. 4.6: информация о команде touch

Используя встроенную переменную $RANDOM, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. 4.7)

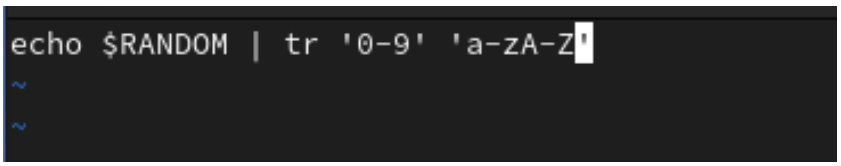

Рис. 4.7: скрипт 3

Запустим данный командный файл и в результате будет выводится несколько букв латинского алфавита рандомно (рис. 4.8)

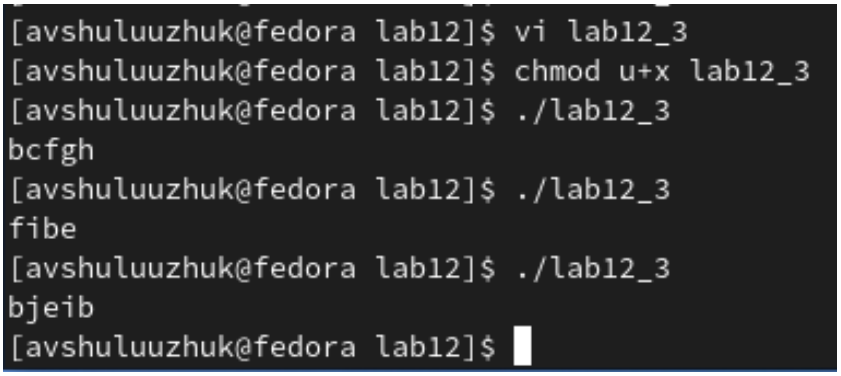

Рис. 4.8: результат командного файла 3

5 Выводы

В ходе выполнения работы мы изучили основы программирования в оболочке ОС UNIX/Linux и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.