Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Латыпова Диана. НФИбд-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 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 Выполнение лабораторной работы

1. Открыла терминал, открыла редактор emacs 12\_1:

**emacs 12\_1**

Написала скрипт к 1 заданию(рис. 1):

lockfile=“./locking.file”

exec {fn}>$lockfile  
  
if test -f "$lockfile"  
then  
while [ 1!=0 ]  
do  
 if flock -n ${fn}  
 then  
 echo "File was locked"  
 sleep 2  
 echo "Unlocking"  
 flock -u ${fn}  
 else  
 echo "File already locked"  
 sleep 2  
 fi  
done

fi

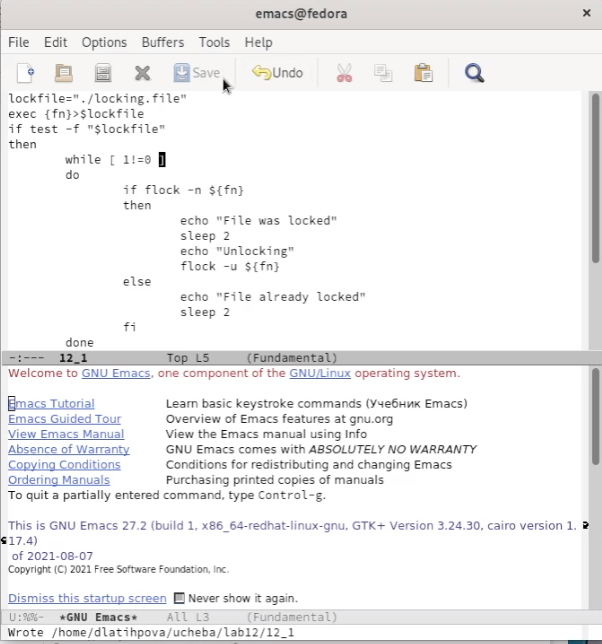


Рис. 1: Скрипт 1 задания

Сначала предоставила права на выполнение файла 12\_1. И запустила файл(рис. 2):

1 **chmod +x 12\_1**

2 **./12\_1**

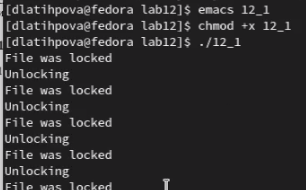


Рис. 2: Результат 1 скрипта

1. Далее через терминал снова открыл редактор emacs, только уже открыла файл 12\_2. Написала там скрипт 2 задания(рис. 3):

command=“”

while getopts :m: opt

do

case $opt in  
 m)command="$OPTARG";;  
 esac

done

if test -f "/usr/share/man/man1/$command.1.gz"  
then less /usr/share/man/man1/$command.1.gz

fi

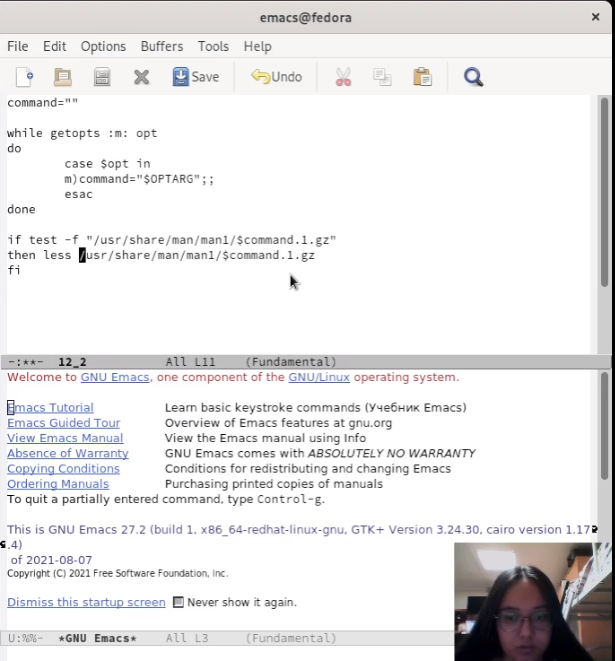


Рис. 3: Скрипт 2 задания

Снова предоставила права на выполнение файла 12\_2. И запустила файл(рис. 4)(рис. 5):

1 **chmod +x 12\_2**

2 **./12\_2 -m ls**

Рис. 4: Результат 2 скрипта(1)

Рис. 4: Результат 2 скрипта(1)

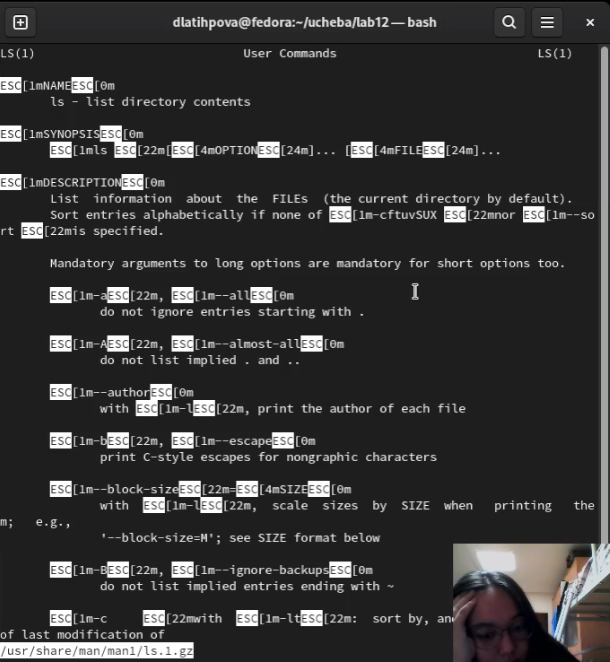


Рис. 5: Результат 2 скрипта(2)

1. И наконец открыла в emacs файл 12\_3, написала там скрипт 3 задания(рис. 6):

* echo "random password: "  
    
  cat /dev/urandom | tr -dc "a-zA-Z0-9" | fold -w 13 | head -n 1

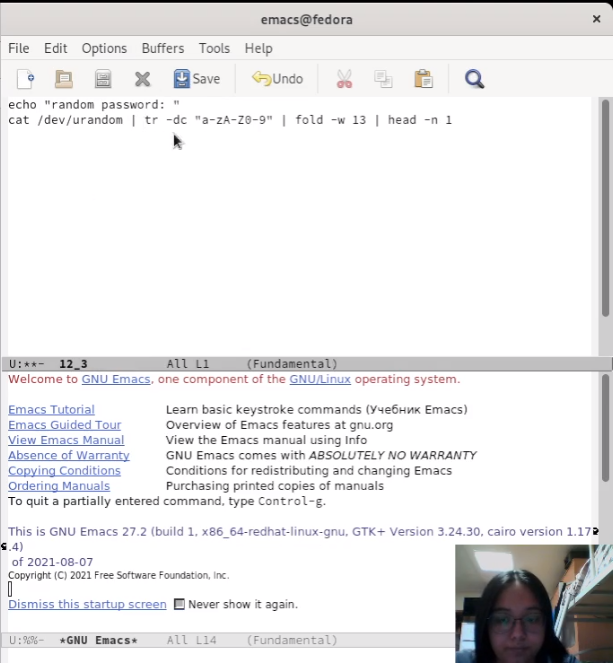


Рис. 6: Скрипт 3 задания

Предоставила права на выполнение файла 12\_. И запустила файл(рис. 7):

1 **chmod +x 12\_3**

2 **./12\_3**

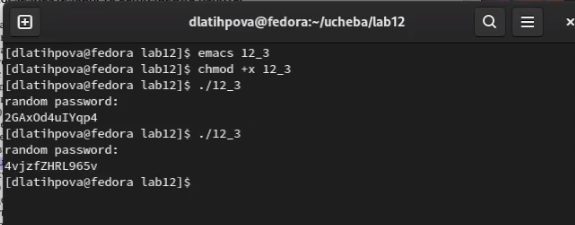


Рис. 7: Результат 3 скрипта

# 4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

* while [$1 != "exit"]

$1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

1. Как объединить (конкатенация) несколько строк в одну?

* cat file.txt | xargs | sed -e 's/. /.\n/g'

1. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

Команда seq выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы.

Команда seq может пригодиться в различных других командах и циклах для генерации последовательности чисел.

Общий синтаксис команды «seq»:

# seq [options] specification

или

Реализовать ее функционал можно командой

for n in {1..5} do <КОМАНДА> done

1. Какой результат даст вычисление выражения $ ((10/3))?

3

1. Укажите кратко основные отличия командной оболочки zsh от bash.

Первое, на что мы взглянем (и это один из наиболее значительных аспектов, на мой взгляд) – это популярность оболочки. Хотя у Z Shell имеется ряд пользователей в среде разработчиков, обычно безопаснее писать свои скрипты для Bash, поскольку гораздо больше людей способны запустить эти скрипты.

Важность всего этого заключается в адаптации скриптов для общедоступных репозиториев, а также в возможности написать грамотную документацию. Благодаря своему большому сообществу для Bash есть несколько крупных ресурсов, которые смогут помочь вам разобраться, как её использовать.

Так что, если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую вам Bash. Хотя это не должно помешать вам использовать Z Shell там, где она более применима для ваших целей. Найти верное решение задачи гораздо важнее, чем взять то, что популярно, так что имейте это в виду.

Хотя Bash куда более распространён, это не означает, что у Z Shell нет полезных возможностей. Её часто хвалят за интерактивную работу, поскольку она лучше настраивается, чем Bash. Например, командная строка более гибкая. Можно отобразить команду слева, а другую справа, как в разделённом экране vim. Автодополнение также быстрее и более изменяемое, чем в Bash.

1. Проверьте, верен ли синтаксис данной конструкции

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

верен

1. 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.

# 5 Выводы

Я изучила основы программирования в оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.