Лабораторная работа №12

Дисциплина: Операционные системы

Кондратьев Арсений Вячеславович

24.09.2022

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 Теоретическое введение

mark - присваивает значение строки символов

let - является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению

break - прерывание циклов

# 3 Выполнение лабораторной работы

1. Написал командный файл, реализующий упрощённый механизм семафоров

Написал скрипт(рис.1)

#!/bin/bash  
  
t1=$1  
t2=$2  
s1=$(date +"%s")  
s2=$(date +"%s")  
((t=$s2-$s1))  
while ((t<t1))  
do  
 echo "Ожидание освобождения"  
 sleep 1  
 s2=$(date +"%s")  
 ((t=$s2-$s1))  
done  
s1=$(date +"%s")  
s2=$(date +"%s")  
((t=$s2-$s1))  
while ((t<t2))  
do  
 echo "Ресурс используется"  
 sleep 1  
 s2=$(date +"%s")  
 ((t=$s2-$s1))  
done

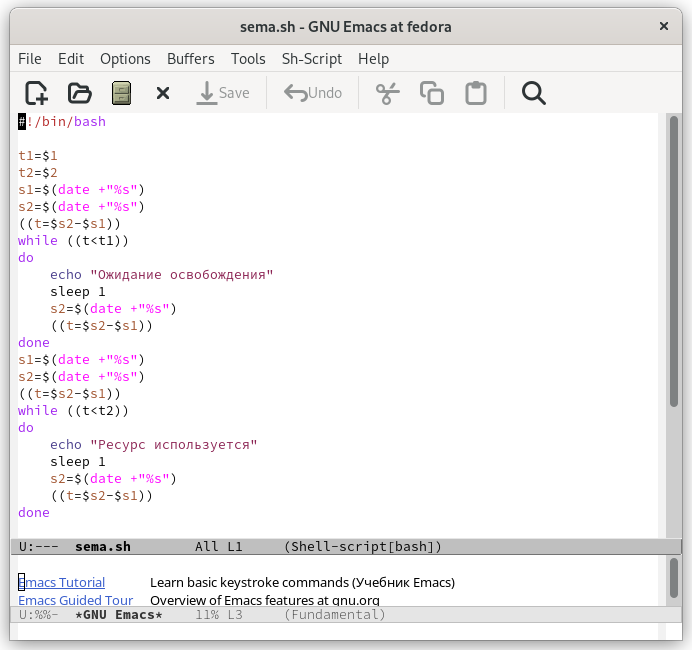


Figure 1: Написанный в Emacs скрипт

В результате с некоторым интервалом получаем сообщения об освобождении, а затем об использовании(рис.2)

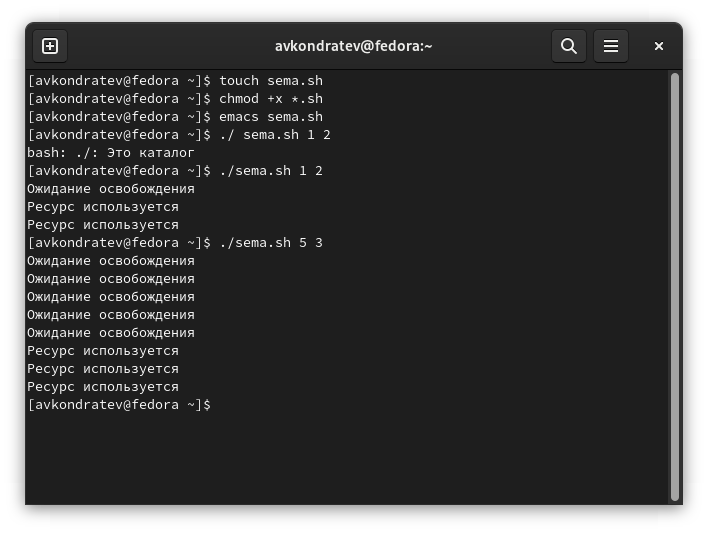


Figure 2: Результат

1. Реализовал команду man с помощью командного файла

Написал скрипт, который разархивирует информацию об введенной команде(рис.3)

#!/bin/bash  
  
command=$1  
if [ -f /usr/share/man/man1/$command.1.gz ]  
then gunzip -c /usr/share/man/man1/$command.1.gz | less  
else echo "Команда не найдена"  
fi

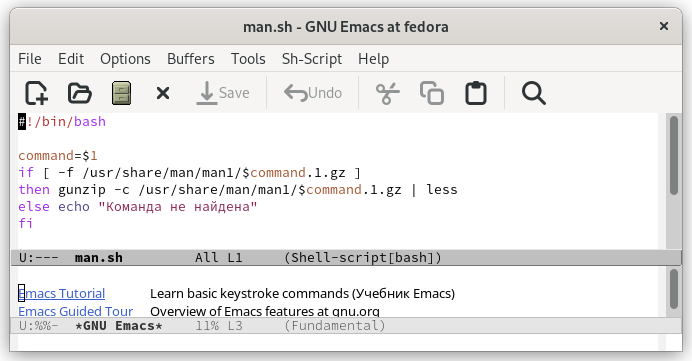


Figure 3: Написанный в Emacs скрипт

В результате получаем описание введенной команды(рис.4)

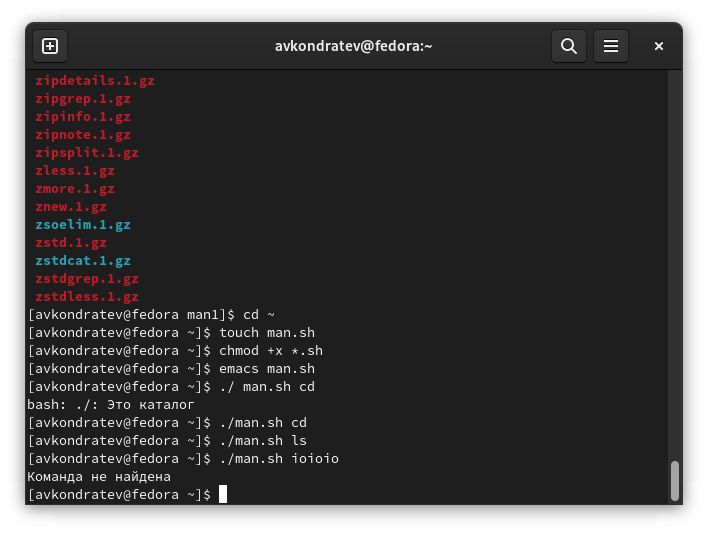


Figure 4: Результат

1. Используя встроенную переменную $RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита

Написал скрипт, который генерирует числа от 1 до 26 и в соответствии с этим числом выводит символ(рис.5)

#!/bin/bash  
  
am=$1  
for ((i=0; i<$am; i++))  
do  
 ((symbol=$RANDOM%26+1))  
 case $symbol in  
 1) echo -n a;; 7) echo -n g;; 12) echo -n l;; 17) echo -n q;; 22) echo -n v;;  
 2) echo -n b;; 8) echo -n h;; 13) echo -n m;; 18) echo -n r;; 23) echo -n w;;  
 3) echo -n c;; 9) echo -n i;; 14) echo -n n;; 19) echo -n s;; 24) echo -n x;;  
 4) echo -n d;; 10) echo -n j;; 15) echo -n o;; 20) echo -n t;; 25) echo -n y;;  
 5) echo -n e;; 11) echo -n k;; 16) echo -n p;; 21) echo -n u;; 26) echo -n z;;  
 6) echo -n f;;  
 esac  
done  
echo

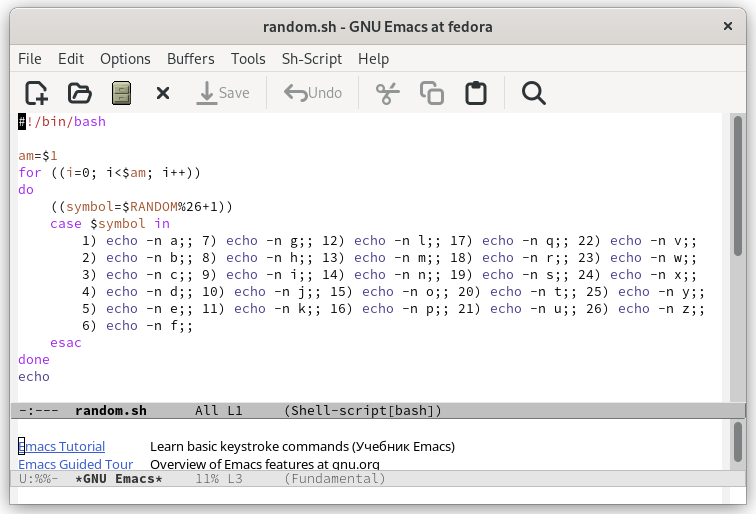


Figure 5: Написанный в Emacs скрипт

В результате получаем случайную последовательность символов(рис.**¿fig:008?**)

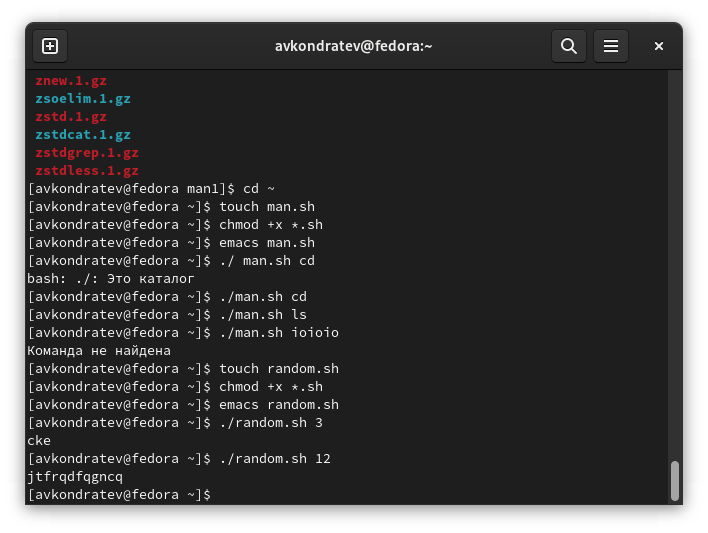


Figure 6: Результат

# 4 Выводы

Я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 5 Контрольные вопросы

1. После открывающей и перед закрывающей квадратными скобками нужно поставить пробел
2. Присвоить двум переменным значения этих строк и в кавычках подряд вывести эти переменные (“str2”)
3. Команда seq выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы.  
   Ее можно реализовать с помощью циклов, например  
   seq 5  
   for ((i=1; i<6; i++))
4. 3
5. Zsh более интерактивный и настраиваемый, чем Bash. У Zsh есть поддержка с плавающей точкой, которой нет у Bash. В Zsh поддерживаются структуры хеш-данных, которых нет в Bash. Функции вызова в Bash лучше по сравнению с Zsh
6. Верен
7. Плюсы:  
   Bash позволяет писать Shell-сценарии с минимальной грамматикой  
   Bash поддерживает процессы нативно  
   Минусы:  
   Bash — это командный язык, а не язык программирования общего назначения. Поэтому с усложнением логики вашего автоматизированного сценария он становится более запутанным и менее читаемым  
   У Bash нет стандартного API, однако он поставляется с простыми встроенными функциями