

Лабораторная работа №12

Дисциплина: Операционные системы

Королев Федор Константинович

Содержание

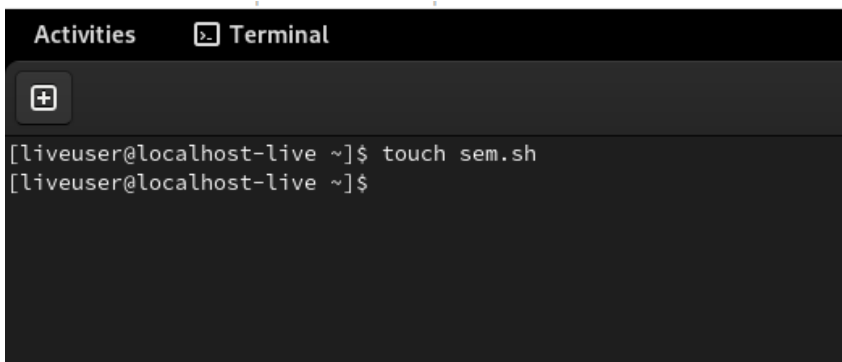
Цель работы	1
Ход работы.....	1
Контрольные вопросы	5
Вывод.....	6

Цель работы

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

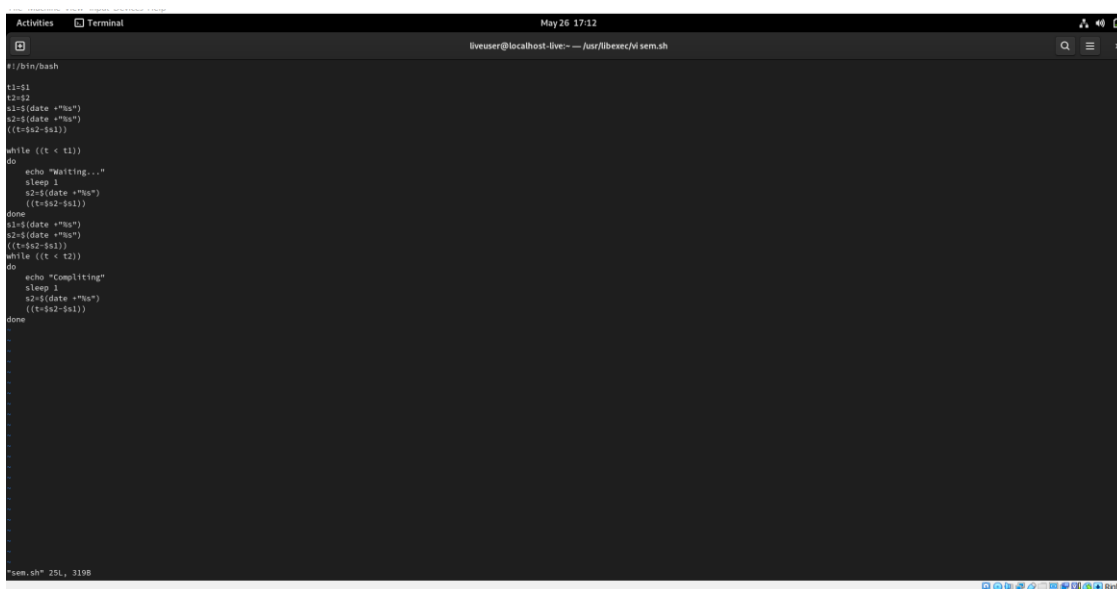
Ход работы

1. Написал командный файл, реализующий упрощенный механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом. Для данной задачи создадим файл `sem.sh` (Рис. 1) и напишем соответствующий скрипт (Рис. 2):



```
Activities  Terminal
[liveuser@localhost-live ~]$ touch sem.sh
[liveuser@localhost-live ~]$
```

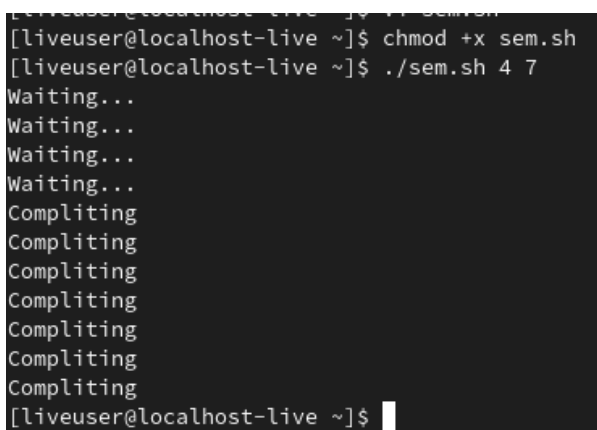
Рис. 1 создание файла



```
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$2-$s1))
while ((t < 1))
do
    echo "Waiting..."
    sleep 1
    s2=$(date +%s)
    ((t=$2-$s1))
done
s3=$(date +%s)
s4=$(date +%s)
((t=$2-$s1))
while ((t < 1))
do
    echo "Compliting"
    sleep 1
    s4=$(date +%s)
    ((t=$2-$s1))
done
```

Рис. 2 написание скрипта

Дадим право на исполнение и проверим работу(Рис. 3):



```
[liveuser@localhost-live ~]$ chmod +x sem.sh
[liveuser@localhost-live ~]$ ./sem.sh 4 7
Waiting...
Waiting...
Waiting...
Waiting...
Compliting
Compliting
Compliting
Compliting
Compliting
Compliting
Compliting
[liveuser@localhost-live ~]$
```

Рис. 3 проверка

2. Реализуем команду man с помощью командного файла. Изучим содержимое каталога /usr/share/man/man1 (Рис. 7, 8). 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.

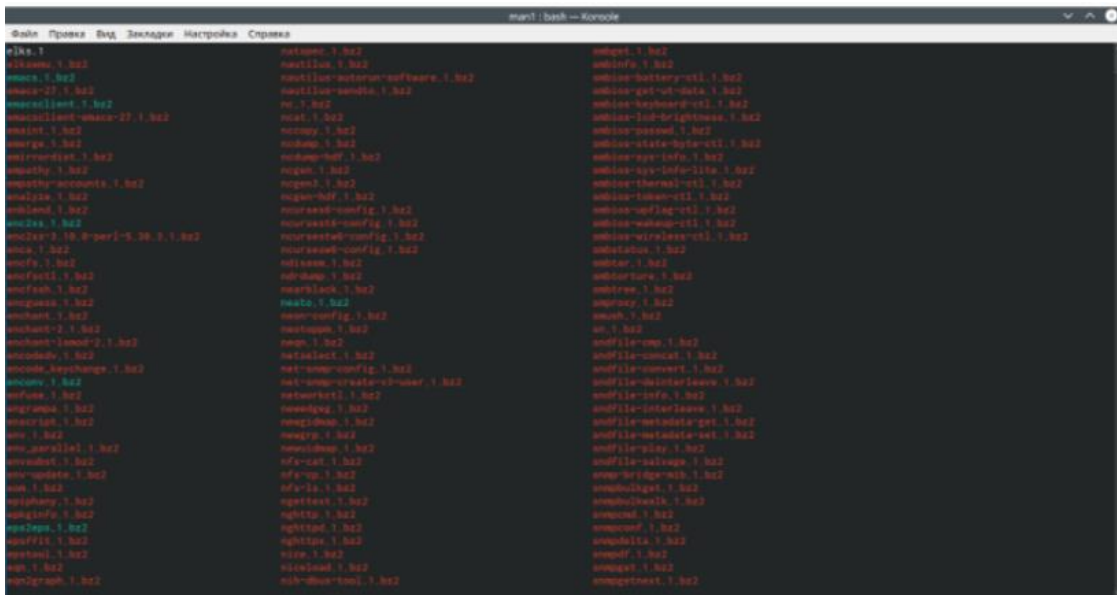


Рис. 4 реализация команды `dpkg-query`

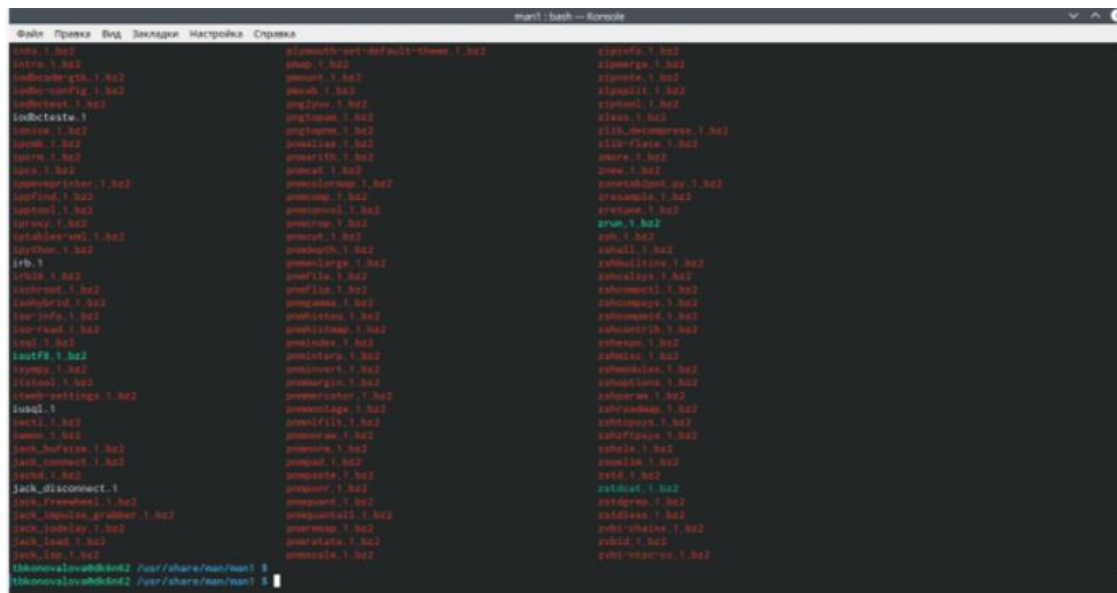


Рис. 5 реализация команды `dpkg-query`

Для реализации создадим файл `man.sh` (Рис. 6, 7):

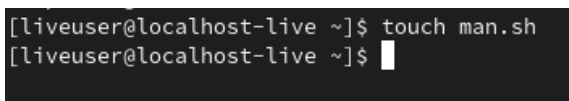


Рис. 6 создание файла

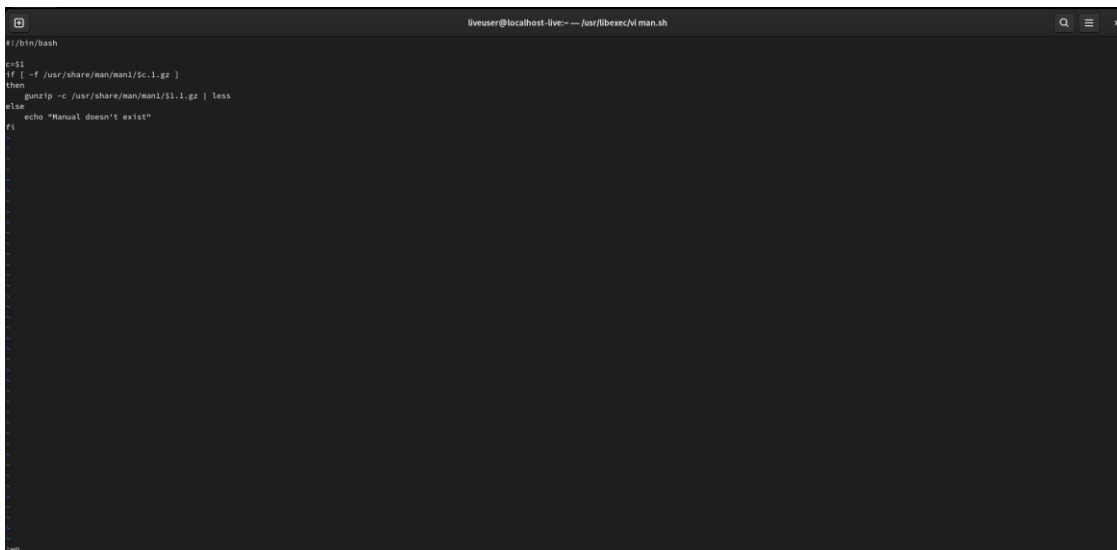


Рис. 7 реализация скрипта

Дадим права доступа и проверим(Рис. 8):

```
[liveuser@localhost-live ~]$ chmod +x man.sh
[liveuser@localhost-live ~]$ ./man.sh mkdir
[liveuser@localhost-live ~]$ ./man.sh ls
```

Рис. 8 проверка

- Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создал файл: random.sh(Рис. 9) и написал соответствующий скрипт(Рис. 10):

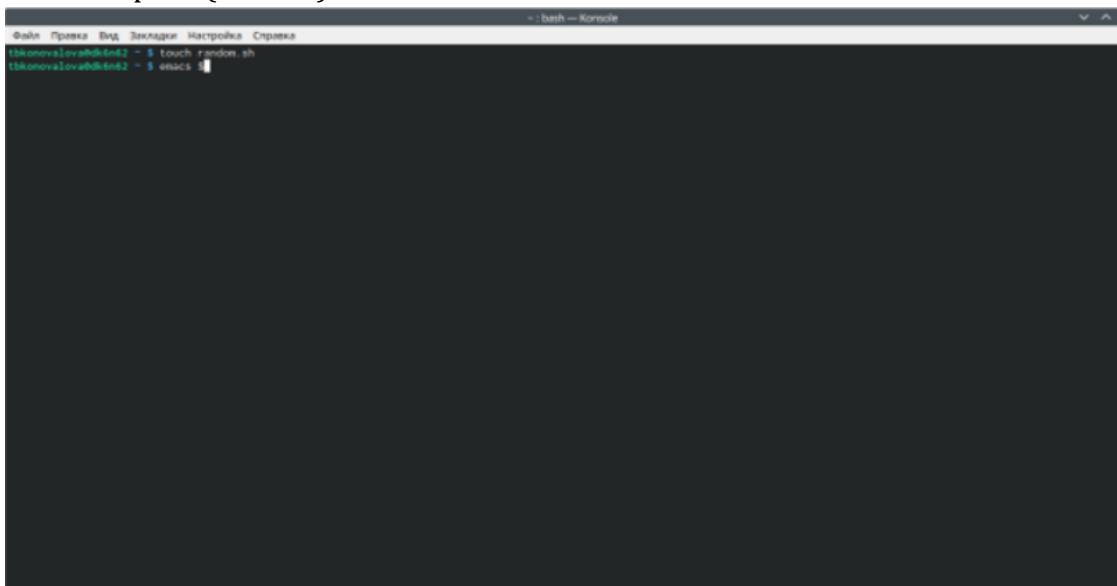


Рис. 9 создание файла

```
#!/bin/bash
i=51
for (( i=0; i<50; i++ ))
do
    (( char=$((RANDOM%26+1)) ))
    case $char in
        1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;; 6) echo -n f;; 7) echo -n g;; 8) echo -n h;; 9) echo -n i;;
        10) echo -n j;; 11) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;;
        21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
    esac
done
echo
```

Рис. 10 скрипт

Проверим работу скрипта, предварительно дав права доступа(Рис. 11):

```
#!/bin/bash
i=51
for (( i=0; i<50; i++ ))
do
    (( char=$((RANDOM%26+1)) ))
    case $char in
        1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;; 6) echo -n f;; 7) echo -n g;; 8) echo -n h;; 9) echo -n i;;
        10) echo -n j;; 11) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;;
        21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
    esac
done
echo
```

Рис. 11 проверка

Контрольные вопросы

1). while [\$1 != "exit"] В данной строчке допущены следующие ошибки: не хватает пробелов после первой скобки [и перед второй скобкой] выражение \$1 необходимо взять в "", потому что эта переменная может содержать пробелы. Таким образом, правильный вариант должен выглядеть так: while ["\$1"!= "exit"] 2). Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: VAR1="Hello," VAR2=" World" VAR3="VAR1VAR2" echo "\$VAR3" Результат: Hello, World Второй: VAR1="Hello," VAR1+=" World" echo "\$VAR1" Результат: Hello, World 3). Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: seq LAST: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение is не выдает. seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT. Если LAST меньше, чем FIRST, он не производит вывод. seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными. seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными. seq -w FIRST INCREMENT LAST: эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными. 4). Результатом данного выражения \$((10/3))будет 3, потому что это целочисленное деление без остатка. 5). Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cdc помощью

Tab В zsh существует калькулятор zcalc, способный выполнять вычисления внутри терминала В zsh поддерживаются числа с плавающей запятой В zsh поддерживаются структуры данных «хэш» В zsh поддерживается раскрытие полного пути на основе неполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim 6). for((a=1; a<= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными (). 7). Преимущества скриптового языка bash: Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS Удобное перенаправление ввода/вывода Большое количество команд для работы с файловыми системами Linux Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка bash: Дополнительные библиотеки других языков позволяют выполнить больше действий Bash не является языком общего назначения Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий.

Вывод

В ходе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.