

Отчет по лабораторной работе №12

дисциплина: Операционные системы

Егорова Юлия Владимировна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Контрольные вопросы	11
4	Выводы	14

Список иллюстраций

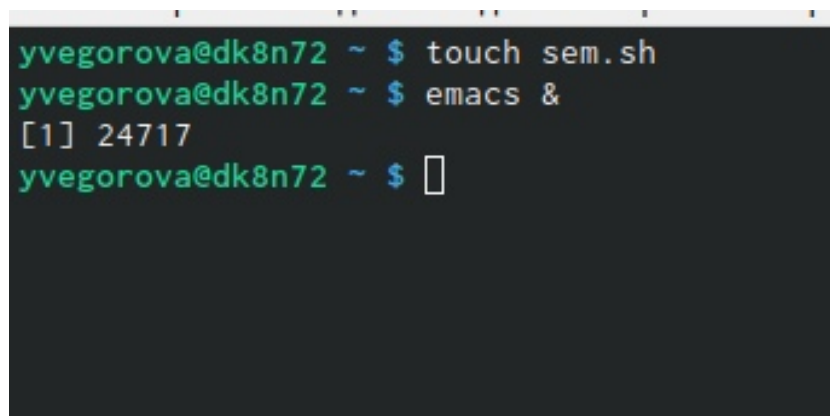
2.1	Создание файла.	5
2.2	Скрипт №1.	6
2.3	Проверка работы программы.	6
2.4	Скрипт №1 (новый).	7
2.5	Проверка работы программы.	7
2.6	Ввод команд.	8
2.7	Вывод содержимого.	8
2.8	Создание файла.	8
2.9	Скрипт №2.	9
2.10	Проверка работы программы.	9
2.11	Создание файла.	9
2.12	Скрипт №3	10
2.13	Проверка работы скрипта №3	10

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

1) Написала командный файл, реализующий упрощённый 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 (процессом). Запустила 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.



```
yvegorova@dk8n72 ~ $ touch sem.sh
yvegorova@dk8n72 ~ $ emacs &
[1] 24717
yvegorova@dk8n72 ~ $
```

Рис. 2.1: Создание файла.

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

U:--- sem.sh All L21 (Shell-script[bash]) Ср мая 25 18:31 1.07
Warning (initialization): An error occurred while loading '~/.emacs':

error: Package 'fira-code-mode-' is unavailable

To ensure normal operation, you should investigate and remove the
cause of the error in your initialization file. Start Emacs with
the '--debug-init' option to view a complete error backtrace.
□

U:%*~ *Warnings* All L8 (Special) Ср мая 25 18:31 1.07
Wrote /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/sem.sh

Рис. 2.2: Скрипт №1.

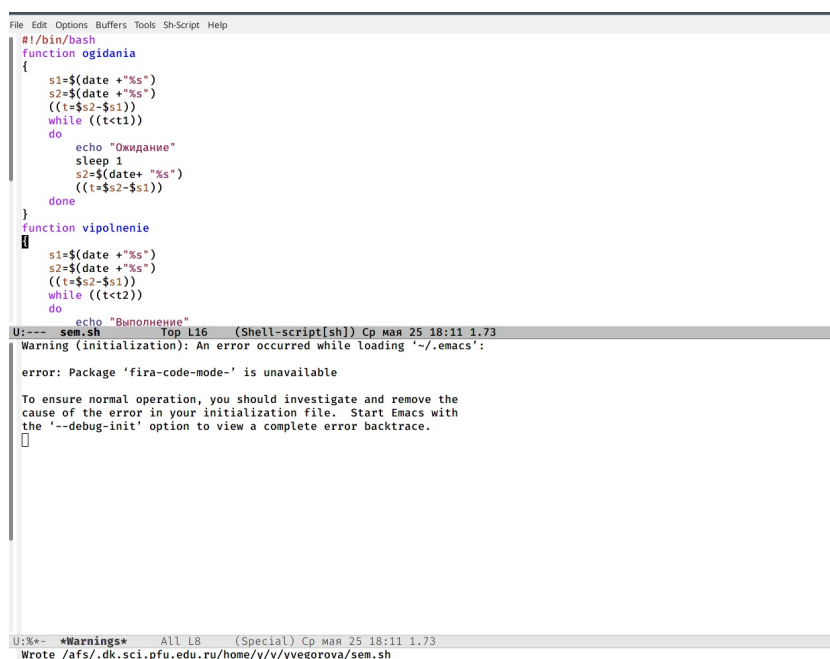
Проверила работу написанного скрипта:

```
yvegorova@dk8n72 ~ $ chmod +x sem.sh
yvegorova@dk8n72 ~ $ ./sem.sh 4 7
Ожидание
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
^Z
[11]+  Остановлен ./sem.sh 4 7
yvegorova@dk8n72 ~ $
```

Рис. 2.3: Проверка работы программы.

Доработала программу так, чтобы имелась возможность взаимодействия

трёх и более процессов:

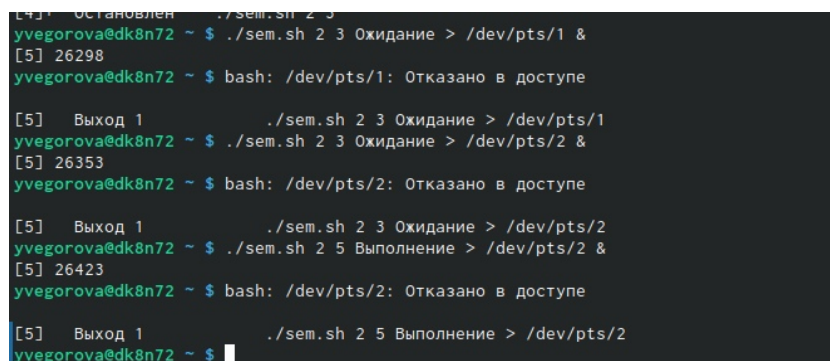


```
#!/bin/bash
function ogidania
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=$s2-$s1))
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=$s2-$s1))
    done
}
function vypolnenie
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=$s2-$s1))
    while ((t<t2))
    do
        echo "Выполнение"
    done
}

# Main script logic
t1=$(date +%s)
t2=$(date +%s)
ogidania
vypolnenie
```

Рис. 2.4: Скрипт №1 (новый).

Проверила работу написанного скрипта:



```
[4] ~ Остановлен ./sem.sh 2 3
yvegorova@dk8n72 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/1 &
[5] 26298
yvegorova@dk8n72 ~ $ bash: /dev/pts/1: Отказано в доступе

[5] Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/1
yvegorova@dk8n72 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/2 &
[5] 26353
yvegorova@dk8n72 ~ $ bash: /dev/pts/2: Отказано в доступе

[5] Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/2
yvegorova@dk8n72 ~ $ ./sem.sh 2 5 Выполнение > /dev/pts/2 &
[5] 26423
yvegorova@dk8n72 ~ $ bash: /dev/pts/2: Отказано в доступе

[5] Выход 1 ./sem.sh 2 5 Выполнение > /dev/pts/2
yvegorova@dk8n72 ~ $
```

Рис. 2.5: Проверка работы программы.

2)Реализововала команду man с помощью командного файла. Изучила 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 ко-

мандной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.

```
yvegorova@dk8n72 ~ $ cd /usr
yvegorova@dk8n72 /usr $ cd share
yvegorova@dk8n72 /usr/share $ cd man
yvegorova@dk8n72 /usr/share/man $ cd man1
yvegorova@dk8n72 /usr/share/man/man1 $ ls
```

Рис. 2.6: Ввод команд.

Рис. 2.7: Вывод содержимого.

Создала файл:

```
yvegorova@dk8n72 /usr/share/man/man1 $ cd ~
yvegorova@dk8n72 ~ $ touch man.sh
yvegorova@dk8n72 ~ $ emacs &
```

Рис. 2.8: Создание файла.



```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
  gunzip -c /usr/share/man/man1/$1.1.gz | less
else
  echo "Справки по данной команде нет"
fi
```

U:-- man.sh All L8 (Shell-script[sh]) Ср мая 25 18:41 0.40
Warning (initialization): An error occurred while loading '~/.emacs':

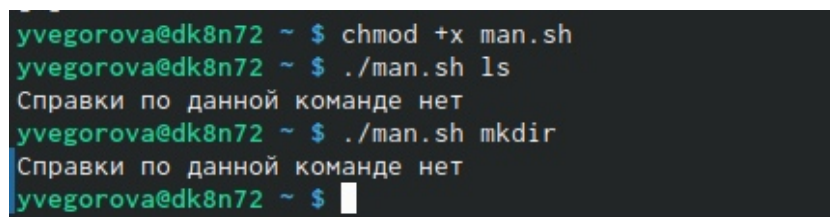
error: Package 'fira-code-mode-' is unavailable

To ensure normal operation, you should investigate and remove the
cause of the error in your initialization file. Start Emacs with
the '--debug-init' option to view a complete error backtrace.
□

U:%*- *Warnings* All L8 (Special) Ср мая 25 18:41 0.40
Wrote /afs/.dk.sci.pfu.edu.ru/home/y/v/yvegorova/man.sh

Рис. 2.9: Скрипт №2.

И проверила работу программы:

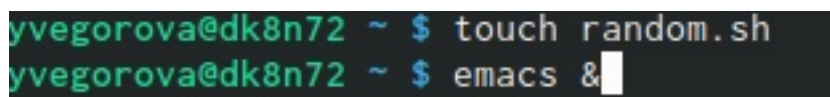


```
yvegorova@dk8n72 ~ $ chmod +x man.sh
yvegorova@dk8n72 ~ $ ./man.sh ls
Справки по данной команде нет
yvegorova@dk8n72 ~ $ ./man.sh mkdir
Справки по данной команде нет
yvegorova@dk8n72 ~ $
```

Рис. 2.10: Проверка работы программы.

3)Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита и учла, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

Для этого я создала файл random.sh и записала соответствующий скрипт:



```
yvegorova@dk8n72 ~ $ touch random.sh
yvegorova@dk8n72 ~ $ emacs &
```

Рис. 2.11: Создание файла.

```
#!/bin/bash
i=5
for (( i=0; i<5; 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
```

U:~: random.sh All L12 (Shell-script[sh]) Co max 25 18:56 0.91

Warning (initialization): An error occurred while loading '~/.emacs':

errors: Package 'fira-code-mode-' is unavailable

To ensure normal operation, you should investigate and remove the cause of the error in your initialization file. Start Emacs with the '--debug-init' option to view a complete error backtrace.

U:~: *Warnings* All L8 (Spec[sh]) Co max 25 18:56 0.91

Wrote /afs/ds.scl.pfe.edu.ru/home/y/y/vegorova/random.sh

Рис. 2.12: Скрипт №3

Далее проверила работу написанного скрипта:

```
yvegorova@dk8n72 ~ $ chmod +x random.sh
yvegorova@dk8n72 ~ $ ./random.sh 7
vrzjiqv
yvegorova@dk8n72 ~ $ ./random.sh 10
nkzfszmvty
yvegorova@dk8n72 ~ $
```

Рис. 2.13: Проверка работы скрипта №3

3 Контрольные вопросы

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.

4 Выводы

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