

# Отчет по лабораторной работе №12

Дисциплина: Операционные системы

Шишук Владислав Олегович

# Содержание

Цель работы	4
Выполнение лабораторной работы	5
Выводы	11
Контрольные вопросы	12

# Список иллюстраций

0.1	рис.1 . . . . .	5
0.2	рис.2 . . . . .	6
0.3	рис.3 . . . . .	6
0.4	рис.5 . . . . .	7
0.5	рис.6 . . . . .	8
0.6	рис.6.1 . . . . .	8
0.7	рис.7 . . . . .	9
0.8	рис.8 . . . . .	9
0.9	рис.9 . . . . .	10

## Цель работы

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

# Выполнение лабораторной работы

1. Создаем файл `ls.sh` и пишем соответствующий скрипт. (рис. -@fig:001) Пишем 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). (рис. -@fig:002)

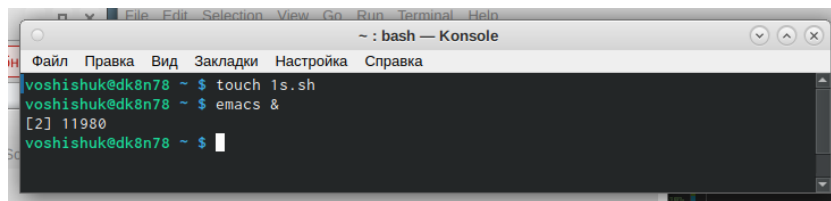


Рис. 0.1: рис.1

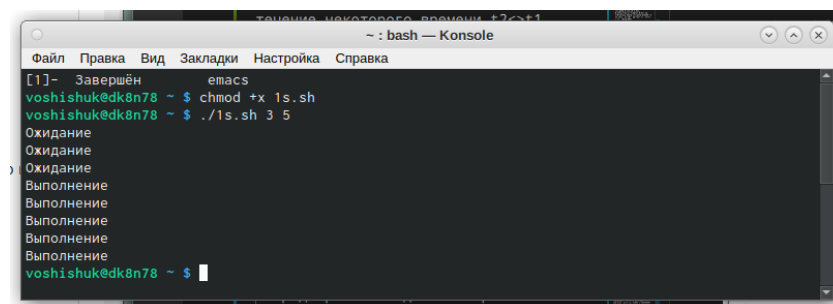


```
#!/bin/bash
t1=$1 #
t2=$2 #
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1)) #
while ((t<t1))
do
    echo "Ожидание"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t<t2))
do
    echo "Выполнение"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))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

U:--- 1s.sh All L10 (Shell-script[sh]) Чт мая 26 14:58 0.22  
Wrote /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/1s.sh

Рис. 0.2: рис.2

Проверяем работу написанного скрипта (команда «./1s.sh 3 5»), предварительно добавив право на исполнение файла (команда «chmod +x 1s.sh»). Скрипт работает корректно.(рис. -@fig:003)



```
[1]- Завершён emacs
voshishuk@dk8n78 ~ $ chmod +x 1s.sh
voshishuk@dk8n78 ~ $ ./1s.sh 3 5
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
voshishuk@dk8n78 ~ $
```

Рис. 0.3: рис.3

После этого изменяем скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверим его работу (команда «./1s.sh 2 5 Ожидание > /dev/pts/2 &» и команда «./1s.sh 2 5 Ожидание > /dev/tty2 »). При этом ни одна из команд не сработала, выводя сообщение “Отказано в до-

ступе”. При этом скрипт работает корректно. (рис. -@fig:004) (рис. -@fig:005)

```

#!/bin/bash
function a
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-$s1)) #
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=s2-$s1))
    done
}
function b
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-$s1))
    while ((t<t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +%s)
        ((t=s2-$s1))
    done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" = "Ожидание" ]
    then a
    fi
    if [ "$command" == "Выполнение" ]
    then b
    fi
    echo "След действие: "
    read command
done
  
```

```

voshishuk@dk8n78 ~$ ./1s.sh 2 5 Ожидание > /dev/pts/2 &
[3] 14069
voshishuk@dk8n78 ~$ bash: /dev/pts/2: Отказано в доступе
./1s.sh 2 5 Ожидание > /dev/pts/2 &
[4] 14102
voshishuk@dk8n78 ~$ ./1s.sh 2 5 Ожидание > /dev/pts/2
[3] Выход 1
voshishuk@dk8n78 ~$ bash: /dev/pts/2: Отказано в доступе
./1s.sh 2 5 Ожидание > /dev/tty2
bash: /dev/tty2: Отказано в доступе
voshishuk@dk8n78 ~$ ./1s.sh 2 5 Ожидание > /dev/pts/2
[4]+ Выход 1
voshishuk@dk8n78 ~$
  
```

Рис. 0.4: рис.5

2. Реализуем команду `map` с помощью командного файла. Изучаем 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`. (рис. -@fig:009)

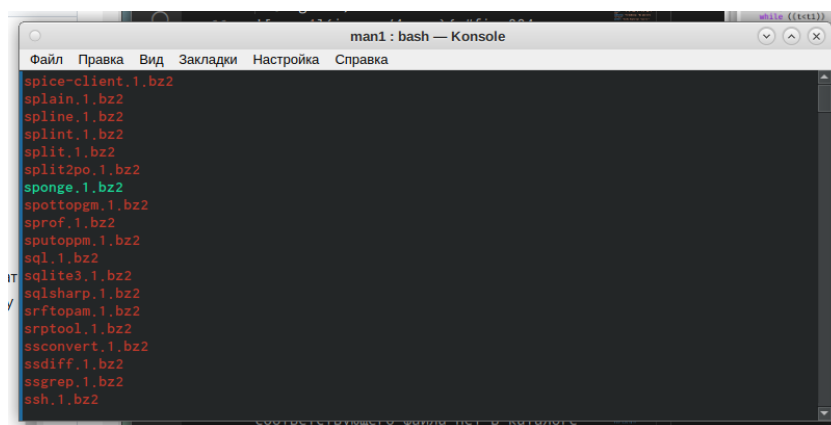


Рис. 0.5: рис.6

Создаем файл `2s.sh` и пишем соответствующие скрипт. (рис. -@fig:007)

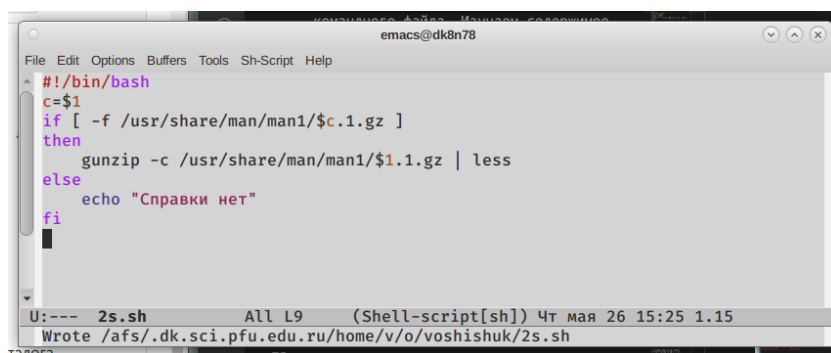
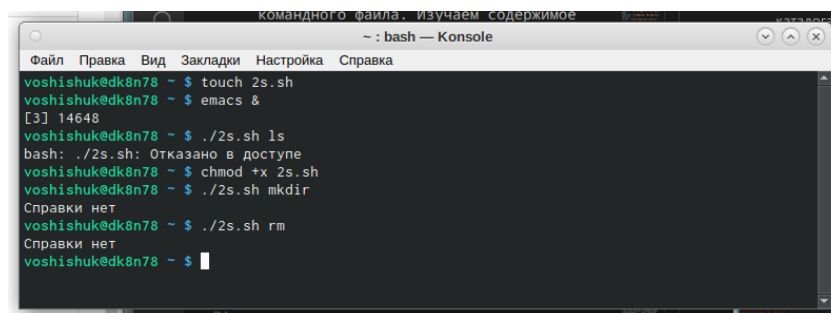


Рис. 0.6: рис.6.1

Проверяем работу написанного скрипта (команды `./2s.sh mkdir` и `./2s.sh rm`),



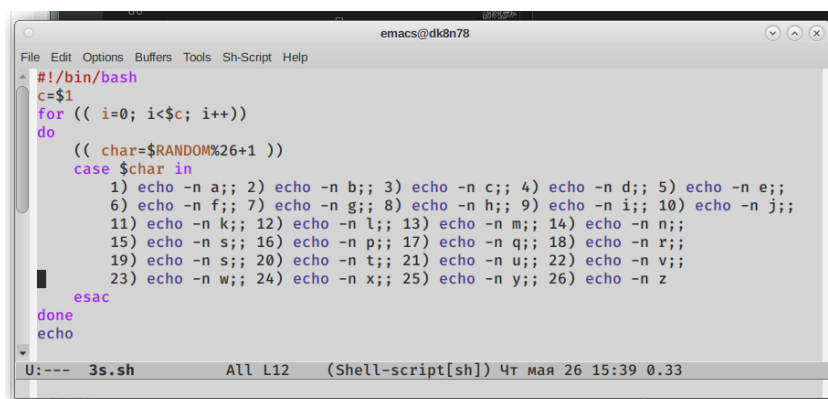
предварительно добавив право на исполнение файла (команда «chmod +x 2s.sh»). Скрипт работает корректно. (рис. -@fig:008)



```
komandnogo fayla. izuchаем содерЖимое
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ touch 2s.sh
voshishuk@dk8n78 ~ $ emacs &
[3] 14648
voshishuk@dk8n78 ~ $ ./2s.sh ls
bash: ./2s.sh: Отказано в доступе
voshishuk@dk8n78 ~ $ chmod +x 2s.sh
voshishuk@dk8n78 ~ $ ./2s.sh mkdir
Справки нет
voshishuk@dk8n78 ~ $ ./2s.sh rm
Справки нет
voshishuk@dk8n78 ~ $
```

Рис. 0.7: рис.7

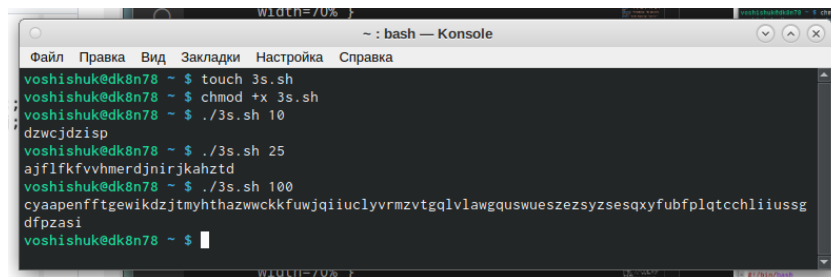
3. Создаем файл 3s.sh и пишем соответствующие скрипты.Используя встроенную переменную \$RANDOM, пишем командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. -@fig:009)



```
emacs@dk8n78
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
c=$1
for (( i=0; i<$c; 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 s;; 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:--- 3s.sh All L12 (Shell-script[sh]) Чт мая 26 15:39 0.33
```

Рис. 0.8: рис.8

Проверяем работу написанного скрипта (команды «./3s.sh 45», «./3s.sh 1000», «./3s.sh 1»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh»). Скрипт работает корректно. (рис. -@fig:010)



A screenshot of a terminal window titled "Konsole" with a menu bar in Russian (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows a user named "voshishuk@dk8n78" performing several commands: "touch 3s.sh", "chmod +x 3s.sh", and ". /3s.sh 10". The script "3s.sh" contains a loop that prints random strings of 25 characters. The output of the script is shown as a long line of random characters: "dzwcjdzisp", "ajflfkfvvhmerdjnirjkahztd", "cyaaapenfftgewikdzjtmvthazwckkfufwjqiuclyvmzvtgqlvlawquswueszezsyzsesqxyfubfplqtcchliussg", and "dfpzasi". The terminal window has a width of 70% and a height of 100%.

```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ touch 3s.sh
voshishuk@dk8n78 ~ $ chmod +x 3s.sh
voshishuk@dk8n78 ~ $ ./3s.sh 10
dzwcjdzisp
voshishuk@dk8n78 ~ $ ./3s.sh 25
ajflfkfvvhmerdjnirjkahztd
voshishuk@dk8n78 ~ $ ./3s.sh 100
cyaaapenfftgewikdzjtmvthazwckkfufwjqiuclyvmzvtgqlvlawquswueszezsyzsesqxyfubfplqtcchliussg
dfpzasi
voshishuk@dk8n78 ~ $
```

Рис. 0.9: рис.9

## Выводы

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

# 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 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 `$((10/3))` будет 3, потому что это целочисленное деление без остатка.
5. Отличия командной оболочки `zsh` от `bash`:
- В `zsh` более быстрое автодополнение для `cd` с помощью `Tab`
  - В `zsh` существует калькулятор `zcalc`, способный выполнять вычисления внутри терминала
  - В `zsh` поддерживаются числа с плавающей запятой
  - В `zsh` поддерживаются структуры данных «хэш»
  - В `zsh` поддерживается раскрытие полного пути на основе неполных данных
  - В `zsh` поддерживается замена части пути
  - В `zsh` есть возможность отображать разделенный экран, такой же как разделенный экран `vim`
6. `for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать `$` 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