

Отчет по лабораторной работе №12

**Программирование в командном процессоре ОС UNIX. Расширенное
программирование**

Галацан Николай, НПИбд-01-22

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Выводы	12
5	Ответы на контрольные вопросы	13

Список иллюстраций

3.1	Запуск программы №1	8
3.2	Запуск программы №2	9
3.3	Справка по команде ls	10
3.4	Запуск программы №3	11

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 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). 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. Написать 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). 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.

Открываю emacs. Создаю файл `lab12_1.sh`, набираю текст программы.

Листинг программы №1:

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

```

        echo "Ожидание освобождения"
        sleep 2
        s2=$(date +%s")
        ((t=$s2-$s1))
done
s1=$(date +%s")
s2=$(date +%s")
while ((t < t2))
do
    echo "Использование"
    sleep 2
    s2=$(date +%s")
    ((t=$s2-$s1))
done

```

Даю файлу право на исполнение и запускаю программу. В качестве первого аргумента передается время на ожидание освобождения файла, в качестве второго - время на исполнение (рис. 3.1).

```

ngalacan@ngalacan:~/work/os/lab12
[ngalacan@ngalacan lab12]$ emacs lab12_1.sh
[ngalacan@ngalacan lab12]$ chmod +x lab12_1.sh
[ngalacan@ngalacan lab12]$ ./lab12_1.sh 3 6
Ожидание освобождения
Ожидание освобождения
Использование
Использование
Использование
[ngalacan@ngalacan lab12]$ ./lab12_1.sh 7 3
Ожидание освобождения
Ожидание освобождения
Ожидание освобождения
Ожидание освобождения
[ngalacan@ngalacan lab12]$

```

Рис. 3.1: Запуск программы №1

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`.

Создаю файл `lab12_2.sh`, набираю текст программы.

Листинг программы №2:

```
#!/bin/bash
cmd=$1
if test -f "/usr/share/man/man1/$cmd.1.gz"
then less /usr/share/man/man1/$cmd.1.gz
else echo "Нет справки по такой команде"
fi
```

Даю файлу право на исполнение и запускаю. В качестве аргумента передаю название команды (рис. 3.2, рис. 3.3).

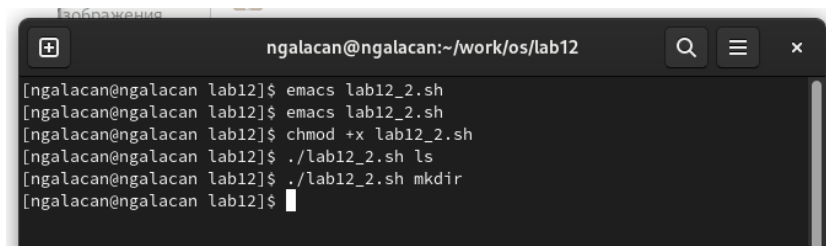


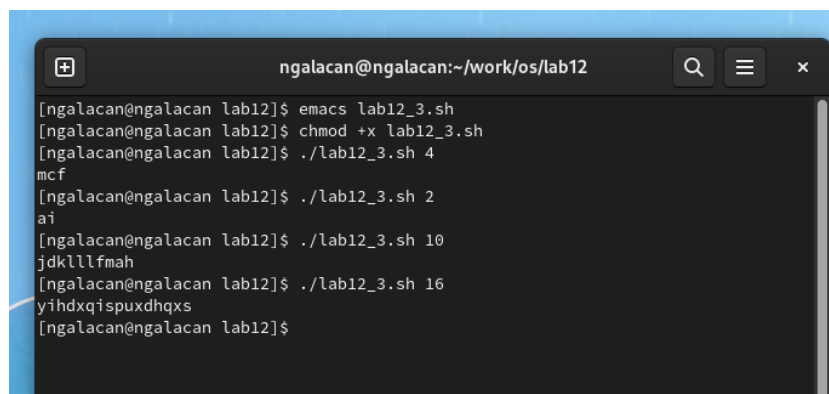
Рис. 3.2: Запуск программы №2


```
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

Даю файлу право на исполнение и запускаю. Значение переменной `letter` может быть от 1 до 26, т.к. остаток от деления `$RANDOM` на 26 не может быть больше 26, и соответствует одной букве латинского алфавита. Выбор осуществляется с помощью `case`. Выводим буквы до тех пор, пока их количество (`i`) меньше введенного аргумента в командной строке (рис. 3.4).



```
ngalacan@ngalacan:~/work/os/lab12  
[ngalacan@ngalacan lab12]$ emacs lab12_3.sh  
[ngalacan@ngalacan lab12]$ chmod +x lab12_3.sh  
[ngalacan@ngalacan lab12]$ ./lab12_3.sh 4  
mcf  
[ngalacan@ngalacan lab12]$ ./lab12_3.sh 2  
ai  
[ngalacan@ngalacan lab12]$ ./lab12_3.sh 10  
jdklllfmah  
[ngalacan@ngalacan lab12]$ ./lab12_3.sh 16  
yihdxqispuxdhqxs  
[ngalacan@ngalacan lab12]$
```

Рис. 3.4: Запуск программы №3

4 Выводы

Изучены основы программирования в оболочке ОС UNIX/Linux. Приобретен навык написания более сложных командных файлов с использованием логических управляющих конструкций и циклов.

5 Ответы на контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

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

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

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

Результатом данного выражения $\$(10/3)$ будет 3, потому что это целочисленное деление без остатка.

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

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. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

Синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

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.