

Лабораторная работа №13

По дисциплине Операционные системы

Выполнил Гамаюнов Н.Е., студент ФФМиЕН РУДН, НПМбд-01-20, 1032201717

Преподаватель Кулябов Дмитрий Сергеевич

Москва, 2021 г.

Цель работы

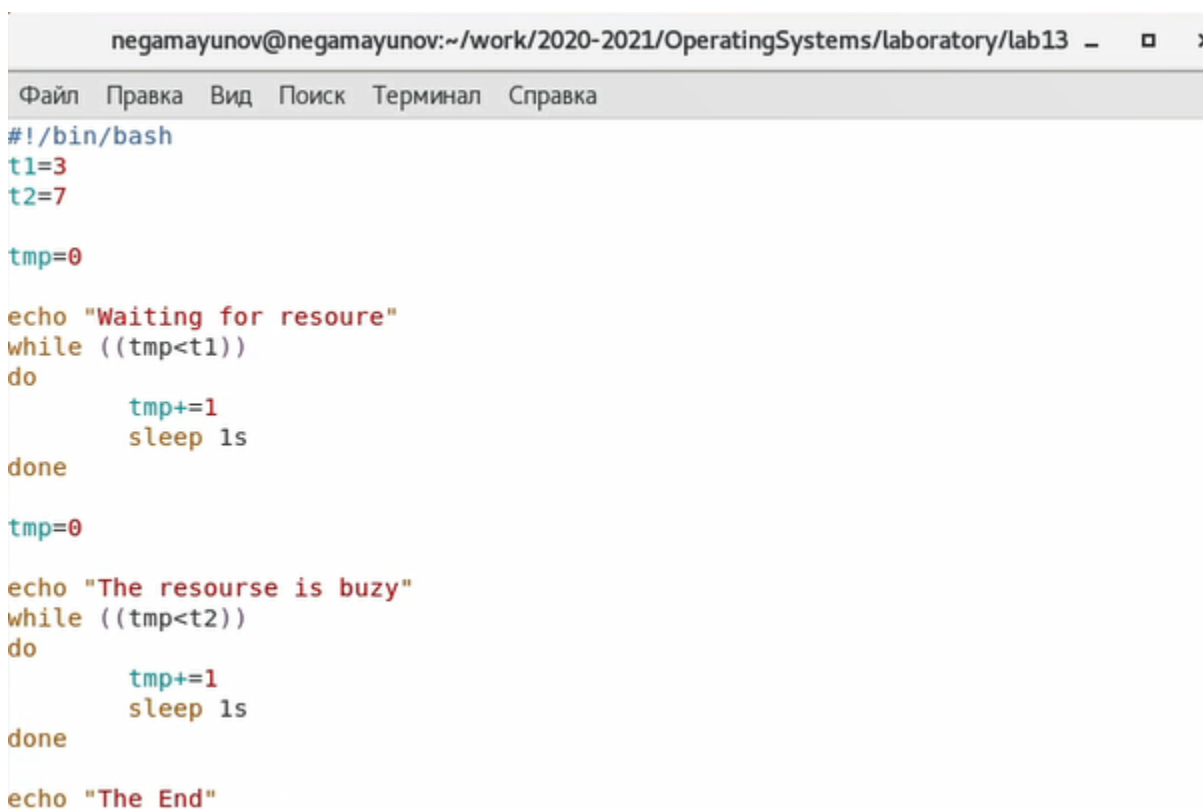
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Задания

1. Написать командный файл, реализующий упрощённый механизм семафоров.
2. Реализовать команду `map` с помощью командного файла.
3. Используя встроенную переменную `$RANDOM`, написать командный файл, генерирующий случайную последовательность букв латинского алфавита.

Выполнение лабораторной работы

1.
 - Написал командный файл, который должен в течение некоторого времени `t1` дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени `t2 <> t1`, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом) (рисунок 1)



```
negamayunov@negamayunov:~/work/2020-2021/OperatingSystems/laboratory/lab13 - □ x
Файл  Правка  Вид  Поиск  Терминал  Справка
#!/bin/bash
t1=3
t2=7

tmp=0

echo "Waiting for resoure"
while ((tmp<t1))
do
    tmp+=1
    sleep 1s
done

tmp=0

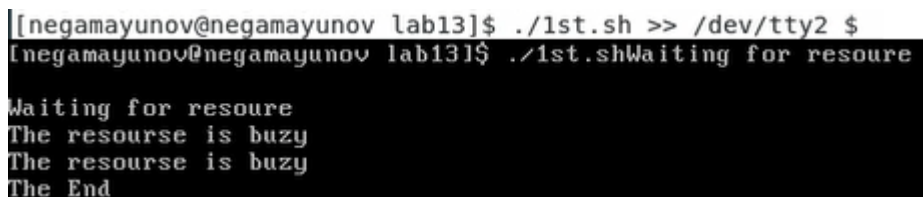
echo "The resourse is buzy"
while ((tmp<t2))
do
    tmp+=1
    sleep 1s
done

echo "The End"
```

Рисунок 1

- Запустил 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 (рисунки 2)



```
[negamayunov@negamayunov lab13]$ ./1st.sh >> /dev/tty2 $
[negamayunov@negamayunov lab13]$ ./1st.shWaiting for resource
Waiting for resource
The resource is busy
The resource is busy
The End
```

Рисунок 2. Сверху - фрагмент графического интерфейса, во второй - терминал tty2. Как мы видим, в терминале результат работы выводился дважды: один раз текст выводил первый процесс, второй - второй.

2. Реализовал команду man с помощью командного файла.

- Изучил содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд (рисунки 3)



```
yum-config-manager.1.gz
yum-debug-dump.1.gz
yum-debug-restore.1.gz
yumdownloader.1.gz
yum-groups-manager.1.gz
yum-utils.1.gz
zcat.1.gz
zcmp.1.gz
zdiff.1.gz
zenity.1.gz
zforce.1.gz
zgrep.1.gz
zip.1.gz
zipcloak.1.gz
zipdetails.1.gz
zipgrep.1.gz
zipinfo.1.gz
zipnote.1.gz
zipsplit.1.gz
zless.1.gz
zmore.1.gz
znew.1.gz
zsoelim.1.gz
[negamayunov@negamayunov man1]$
```

Рисунок 3

Каждый архив можно открыть командой less сразу же просмотрев содержимое справки. Командный файл получает в виде аргумента командной строки название команды и в виде результата выдаёт справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1.

В методических материалах написано, что архив можно сразу открыть командой less. Однако, когда я попытался это сделать, вывод содержал множество текстовых артефактов, что делало справку практически нечитаемой (рисунки 4). Поэтому я решил предварительно разархивировать архивы с помощью команды gunzip с ключом -s (позволяющим не удалять исходный архив) (рисунки 5). Результат выполнения команды ./2nd.sh man показан на рисунке 6.

```

negamayunov@negamayunov:~/work/2020-2021/OperatingSystems/laboratory/lab13 - □ ×
Файл Правка Вид Поиск Терминал Справка

ESC[1mNAMEESC[0m
    man - an interface to the on-line reference manuals

ESC[1mSYNOPSISESC[0m
    ESC[1mman ESC[22m[ESC[1m-C ESC[4mESC[22mfileESC[24m] [ESC[1m-dESC[22m]
    [ESC[1m-DESC[22m] [ESC[1m--warningsESC[22m[=ESC[4mwarningsESC[24m]] [ESC[1m-
R ESC[4mESC[22mencodingESC[24m] [ESC[1m-LESC[0m
    ESC[4mlocaleESC[24m] [ESC[1m-m ESC[4mESC[22msystemESC[24m[,...]] [ESC[1m-
M ESC[4mESC[22mpathESC[24m] [ESC[1m-S ESC[4mESC[22mlistESC[24m] [ESC[1m-e ESC[
4mESC[22mextensionESC[24m] [ESC[1m-iESC[22m]ESC[1m-IESC[22m]
    [ESC[1m--regexESC[22m]ESC[1m--wildcardESC[22m] [ESC[1m--names-onlyESC[2
2m] [ESC[1m-aESC[22m] [ESC[1m-uESC[22m] [ESC[1m--no-subpagesESC[22m] [ESC[1m
-PESC[0m
    ESC[4mpagerESC[24m] [ESC[1m-r ESC[4mESC[22mpromptESC[24m] [ESC[1m-7ESC[22
m] [ESC[1m-E ESC[4mESC[22mencodingESC[24m] [ESC[1m--no-hyphenationESC[22m] [ESC[
1m--no-justifi-ESC[0m
    ESC[1m-cationESC[22m] [ESC[1m-p ESC[4mESC[22mstringESC[24m] [ESC[1m-t
ESC[22m] [ESC[1m-TESC[22m[ESC[4mdeviceESC[24m]] [ESC[1m-HESC[22m[ESC[4mbrowser
ESC[24m]] [ESC[1m-XESC[22m[ESC[4mdpiESC[24m]] [ESC[1m-ZESC[22m]
    [[ESC[4msectionESC[24m] ESC[4mpageESC[24m ...] ...
    ESC[1mman -k ESC[22m[ESC[4maproposESC[24m ESC[4moptionsESC[24m] ESC[4mreg
:

```

Рисунок 4

```

#!/bin/bash
name=$1
if test -f /usr/share/man/man1/$name.1.gz
then gunzip -c /usr/share/man/man1/$name.1.gz | less
else echo "Man does not exist"
fi

```

Рисунок 5

```

negamayunov@negamayunov:~/work/2020-2021/OperatingSystems/laboratory/lab13 - □ ×
Файл Правка Вид Поиск Терминал Справка

.RB $ LANG
or another system dependent environment variable to your language locale,
usually specified in the
.B POSIX 1003.1
based format:

.\
.\ " Need a \c to make sure we don't get a space where we don't want one
.\
.RI < language >[\|\c
.B _\c
.RI < territory >[\|\c
.B .\c
.RI < character-set >[\|\c
.B ,\c
.RI < version >[\|\|\|\]

If the desired page is available in your
.IR locale ,
it will be displayed in lieu of the standard
(usually American English) page.

Support for international message catalogues is also featured in this
:

```

Рисунок 6. Хотя вывод и не идентичен выводу map, информация о команде читаема, её можно понять.

3. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Принцип работы очень прост: мы получаем случайную длину строки (от 1 до 10), затем получаем номер для каждой буквы, соответствующий номеру этой буквы относительно начала латинской алфавита (начиная с нуля). И с помощью оператора выбора case выводим нужную нам букву (*рисунок 7*) Результат выполнения - *рисунок 8*.

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

Рисунок 7

```
[negamayunov@negamayunov lab13]$ ./3rd.sh
x
p
v
a
g
w
g
a
a
```

Рисунок 8

Выводы

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

Контрольные вопросы

1. \$1 необходимо взять в кавычки, чтобы учесть пробелы в записи.
2. Самый простой способ объединить две или более строковых переменных — записать их одну за другой: VAR1="Hello,"

```
VAR2=" World"
```

```
VAR3="$VAR1$VAR2"
```

```
echo "$VAR3"
```

Последняя строка будет отображать объединенную строку:

```
Hello, World
```

Источник routerus.com | [Конкатенация строк в Bash](#)

3. Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT.

Синтаксис:

```
seq [OPTION]... LAST
```

```
или seq [OPTION]... FIRST LAST
```

```
или seq [OPTION]... FIRST INCREMENT LAST
```

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.

Источник espressocode.top | команда `seq`

В `bash` работу этой команды можно реализовать с помощью `for` или `while`.

4. 3

- 5.
 - 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`

Источник - habr.com | `zsh` и `bash`: что выбрать

6. Верен

- 7. Из плюсов стоит отметить, что `bash` установлен по умолчанию на многих дистрибутивах `MacOS` и `Unix`, довольно прост в освоении и обладает мощным арсеналом команд для манипуляций с файловыми системами. А ещё есть возможность упрощать работу с помощью самописных скриптов.

К недостаткам же я бы отнес тот факт, что `bash`, всё-таки, не является языком общего назначения, и его скрипты нельзя без манипуляций запустить на других операционных системах. Да и родные утилиты порой замедляют работу скриптов, запуская много новых процессов.

Библиография

- [Кулябов Д. С. и др. Операционные системы. Методические рекомендации к лабораторной работе №11](#)
- [Кулябов Д. С. и др. Операционные системы. Методические рекомендации к лабораторной работе №13](#)
- routerus.com | Конкатенация строк в `Bash`
- habr.com | `zsh` и `bash`: что выбрать