

Отчёт по лабораторной работе №12

Дисциплина: Операционные системы

Дупленских Василий Викторович

Содержание

Цель работы:	5
Выполнение лабораторной работы:	6
1.1. Создаю новый текстовый файл semafor.sh и запускаю emacs:	6
1.2. Пишу скрипт который при запуске будет показывать упрощенный механизм семафора:	7
2.1. Создаю новый текстовый файл map.sh и запускаю emacs:	7
2.2. Пишу скрипт который показывает справку по введенной команде: . .	9
3.1. Создаю новый текстовый файл random.sh и запускаю emacs:	10
3.2. Проверяю третий скрипт:	10
Ответы на вопросы:	11
Вывод:	14

Список иллюстраций

0.1	Создание текстового файла	6
0.2	Скрипт 1 работа	7
0.3	Создание текстового файла	7
0.4	Скрипт 2 работа	9
0.5	Создание текстового файла	10
0.6	Скрипт 3 работа	10

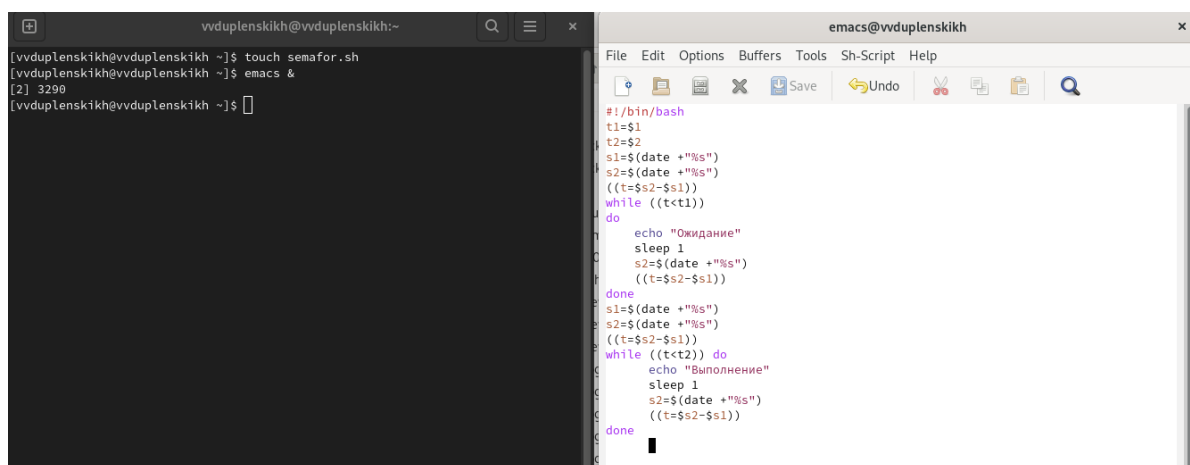
Список таблиц

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:

1.1. Создаю новый текстовый файл semafor.sh и запускаю emacs:



```
[vvduplenskikh@vvduplenskikh ~]$ touch semafor.sh
[vvduplenskikh@vvduplenskikh ~]$ emacs &
[2] 3298
[vvduplenskikh@vvduplenskikh ~]$
```

```
#!/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
```

Рис. 0.1: Создание текстового файла

1.2. Пишу скрипт который при запуске будет показывать упрощенный механизм семафора:

```
[vvduplenskikh@vvduplenskikh ~]$ chmod +x semafor.sh
[vvduplenskikh@vvduplenskikh ~]$ ./semafor.sh 3 5
Ожидание
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
[vvduplenskikh@vvduplenskikh ~]$
```

Рис. 0.2: Скрипт 1 работа

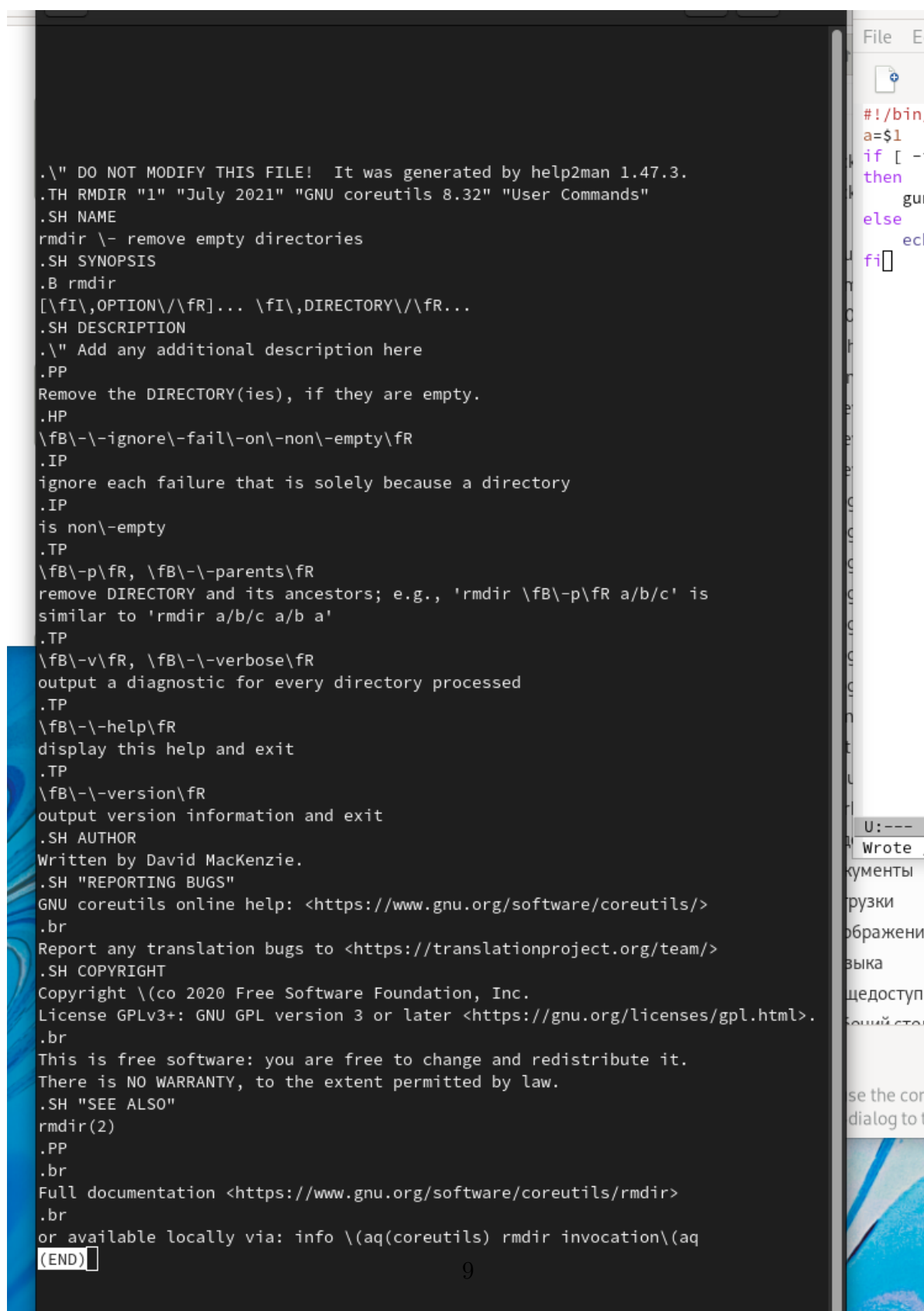
2.1. Создаю новый текстовый файл man.sh и запускаю emacs:

```
[vvduplenskikh@vvduplenskikh ~]$ touch man.sh
[vvduplenskikh@vvduplenskikh ~]$
```

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

Рис. 0.3: Создание текстового файла

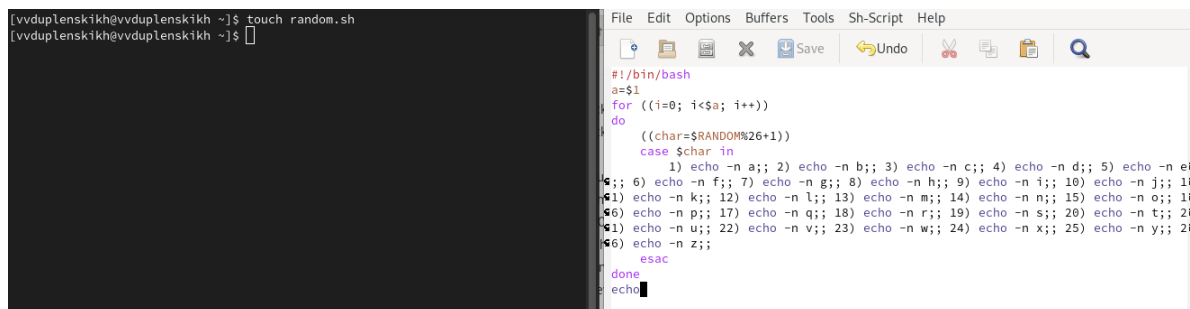
2.2. Пишу скрипт который показывает справку по введенной команде:



```
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH RMDIR "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
rmdir \- remove empty directories
.SH SYNOPSIS
.B rmdir
[\fI\,OPTION\|\fR]... \fI\,DIRECTORY\|\fR...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Remove the DIRECTORY(ies), if they are empty.
.HP
\fB\--ignore\--fail\--on\--non\--empty\|fR
.IP
ignore each failure that is solely because a directory
.IP
is non\--empty
.TP
\fB\--p\|fR, \fB\--\--parents\|fR
remove DIRECTORY and its ancestors; e.g., 'rmdir \fB\--p\|fR a/b/c' is
similar to 'rmdir a/b/c a/b a'
.TP
\fB\--v\|fR, \fB\--\--verbose\|fR
output a diagnostic for every directory processed
.TP
\fB\--help\|fR
display this help and exit
.TP
\fB\--version\|fR
output version information and exit
.SH AUTHOR
Written by David MacKenzie.
.SH "REPORTING BUGS"
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
.br
Report any translation bugs to <https://translationproject.org/team/>
.SH COPYRIGHT
Copyright \(\co 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
.br
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
.SH "SEE ALSO"
rmdir(2)
.PP
.br
Full documentation <https://www.gnu.org/software/coreutils/rmdir>
.br
or available locally via: info \(\aq(coreutils) rmdir invocation\(\aq
(END)
```

Рис. 0.4: Скрипт 2 работа

3.1. Создаю новый текстовый файл random.sh и запускаю emacs:

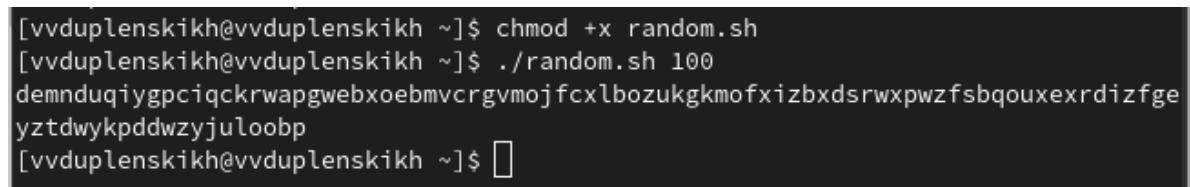


```
[vvduplenskikh@vvduplenskikh ~]$ touch random.sh
[vvduplenskikh@vvduplenskikh ~]$

#!/bin/bash
a=1
for ((i=0; i<$a; i++))
do
  ((char=$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)
    12) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16)
    17) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;; 21)
    22) 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. 0.5: Создание текстового файла

3.2. Проверяю третий скрипт:



```
[vvduplenskikh@vvduplenskikh ~]$ chmod +x random.sh
[vvduplenskikh@vvduplenskikh ~]$ ./random.sh 100
demnduqi ygpciqckrwapgwebxoebm vcr gvmojfcx lboz uk gkm ofxizbx dsrwxpwzfsbqouxexrdizfge
yztdwykpddwzyjuloobp
[vvduplenskikh@vvduplenskikh ~]$
```

Рис. 0.6: Скрипт 3 работа

Ответы на 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

Вывод:

Я изучил основы программирования в оболочке ОС UNIX. Также Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.