

# Отчет

Лабораторная работа №12

Павлова Варвара Юрьевна НПМбд-02-21

# Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Выполнение лабораторной работы	8
Выводы	15
Список литературы	16

## Список иллюстраций

0.1	создание файла . . . . .	8
0.2	написание скрипта . . . . .	9
0.3	выполнение скрипта . . . . .	10
0.4	изменение скрипта . . . . .	10
0.5	изменение скрипта . . . . .	11
0.6	проверка первого файла . . . . .	11
0.7	переход в каталог . . . . .	11
0.8	создание файла . . . . .	12
0.9	написание скрипта . . . . .	12
0.10	проверка второго файла . . . . .	12
0.11	вывод справки . . . . .	13
0.12	создание файла . . . . .	13
0.13	написание скрипта . . . . .	14
0.14	проверка третьего файла . . . . .	14

## Список таблиц

## Цель работы

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

# Задание

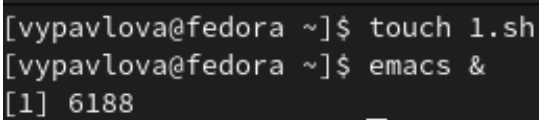
1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию 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 имела 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.

# Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; - C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

# Выполнение лабораторной работы

1. Создаю первый исполняемый файл 1.sh и открываю редактор emacs. (рис. [-@fig:001])



```
[vypavlova@fedora ~]$ touch 1.sh
[vypavlova@fedora ~]$ emacs &
[1] 6188
```

Рис. 0.1: создание файла

2. Пишу 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]) Добавляю право на выполнение и проверяю работу файла. (рис. [-@fig:003])



```
#!/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. 0.2: написание скрипта

```
[vypavlova@fedora ~]$ chmod +x 1.sh
[vypavlova@fedora ~]$ ./1.sh 2 4
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
```

Рис. 0.3: выполнение скрипта

3. Модифицирую и запускаю 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. (рис. [-@fig:004]) (рис. [-@fig:005]) (рис. [-@fig:006])

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

Рис. 0.4: изменение скрипта

```

t1=$1
t2=$2
command=$3
while true
do
    if ["$command" == "exit"]
    then
        echo "exit"
        exit 0
    fi
    if ["$command" == "ожидание"]
    then o
    fi
    if ["$command" == "выполнение"]
    then v
    fi
    echo "next: "
    read command
done

```

Рис. 0.5: изменение скрипта

```

[vypravlova@fedora ~]$ ./1.sh 2 4 ожидание
Ожидание
Ожидание
next:
выполнение
Выполнение
Выполнение
Выполнение
Выполнение
next:
exit
exit

```

Рис. 0.6: проверка первого файла

4. Изучаю содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд.(рис. [-@fig:007])

```

[vypravlova@fedora ~]$ cd usr/share/man/man1
bash: cd: usr/share/man/man1: No such file or directory

```

Рис. 0.7: переход в каталог

5. Создаю второй исполняемый файл 2.sh и открываю редактор emacs. (рис. [-@fig:008])

```
[vypavlova@fedora ~]$ touch 2.sh
[vypavlova@fedora ~]$ emacs &
[1] 7925
```

Рис. 0.8: создание файла

6. Пишу командный файл, который 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])

```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/${c}.1.gz ]
then
    gunzip -c /usr/share/man/man1/${c}.1.gz | less
else
    echo "no info"
fi
```

Рис. 0.9: написание скрипта

7. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:010])  
(рис. [-@fig:011])

```
[vypavlova@fedora ~]$ chmod +x 2.sh
[vypavlova@fedora ~]$ ./2.sh ls
```

Рис. 0.10: проверка второго файла

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH LS "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[\fI\,OPTION\[\fR]... [\fI\,FILE\[\fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB\-\cftuvSUX\[\fR nor \fB\-\sort\[\fR is sp
ecified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-\a\[\fR, \fB\-\-all\[\fR
do not ignore entries starting with .
.TP
\fB\-\A\[\fR, \fB\-\-almost\-\all\[\fR
do not list implied . and ..
.TP
\fB\-\-author\[\fR
:

```

Рис. 0.11: вывод справки

8. Создаю третий исполняемый файл 3.sh и открываю редактор emacs. (рис. [-@fig:012])

```

[vypavlova@fedora ~]$ touch 3.sh
[vypavlova@fedora ~]$ emacs &
[1] 8681

```

Рис. 0.12: создание файла

9. Используя встроенную переменную \$RANDOM, пишу командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. [-@fig:013])

```
#!/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 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
```

Рис. 0.13: написание скрипта

10. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:014])

```
[vypavlova@fedora ~]$ chmod +x 3.sh
[vypavlova@fedora ~]$ ./3.sh 43
dvexvwdstojtfinrxwyvvlhnyiahtekuzynohnwogak
[vypavlova@fedora ~]$ ./3.sh 430
hrjthwvobppzylauvocomhsvnunthkvnfabbwokuhxuiifmorednapkbvfnkdnfsakjumlcflwopzy
mkdgmvtowjdzehovucjtlmopdhodxepbrdqvxmxsyskhfebslxqdusnumtjmchkdqrxzgnrpqwrpn
jmahwnmtgmympuiatvdckvxscudrqhopshfyrfaufvugcycwmavrsquwylxxfoxkpxbnlmajvfqsqxw
srxjttzibkyryimrzgzhkefjmjqfquvdgxxopjytxpkpakzcfziraboxqtsgiatvomxcqsjejkrr
nzziooeefmodzowdudkmhoqsagflccyygsniitqmgyvqbknzqxwsjnewfmxgmglczndydgoclkpntov
wzyocjuvqdxhpbhcnvcjcthmxfifh
```

Рис. 0.14: проверка третьего файла

## Выводы

Выполняя данную лабораторную работу я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## Список литературы