

Лабораторная работа №14

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

Кижваткина Анна Юрьевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11

Список иллюстраций

2.1	Создание директории	6
2.2	Создание файла	6
2.3	Программа	7
2.4	Предоставление доступа	7
2.5	Запуск программы	8
2.6	Создание файла	8
2.7	Программа	9
2.8	Запуск программы	9
2.9	Создание файла	10
2.10	Программа	10

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Создаем директорию lab14 и перемещаемся в неё. (рис. 2.1)

```
[aykizhvatkina@vbox ~]$ mkdir lab14  
[aykizhvatkina@vbox ~]$ cd lab14  
[aykizhvatkina@vbox lab14]$
```

Рис. 2.1: Создание директории

Создаем файл 1.sh. (рис. 2.2)

```
[aykizhvatkina@vbox lab14]$ touch 1.sh
```

Рис. 2.2: Создание файла

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). 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. . (рис. 2.3)

```

1.sh [-M--] 0 L:[ 1+24 25/ 28] *(569 / 598b) 0114 0[*][x
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Использование: $0 <t1> <t2>"
    exit 1
fi

t1=$1
t2=$2

semaphore_file="semaphore.lock"

touch $semaphore_file

function access_resource {
    while ! touch $semaphore_file $0.lock 2>/dev/null; do
        <----->echo "Процесс ждет, ожидание окончено..."
        <----->sleep $t1
    done

    echo "Процесс окончено, теперь использование на $t2 секунд"
    sleep $t2
    echo "Процесс окончено, использование завершено"
}

touch $0.lock
}

access_resource

```

Рис. 2.3: Программа

Устанавливаем право на выполнение. (рис. 2.4)

```

[aykizhvatkina@vbox lab14]$ chmod +x 1.sh

```

Рис. 2.4: Предоставление доступа

Проверяем выполнение программы. (рис. 2.5)

```
bash: ./1.sh: Отказано в доступе
[aykizhvatkina@vbox lab14]$ chmod +x 1.sh
[aykizhvatkina@vbox lab14]$ ./1.sh
Использование: ./1.sh <t1> <t2>
[aykizhvatkina@vbox lab14]$ mc

[aykizhvatkina@vbox lab14]$ ./1.sh
Использование: ./1.sh <t1> <t2>
[aykizhvatkina@vbox lab14]$ mc

[aykizhvatkina@vbox lab14]$ ./1.sh
Использование: ./1.sh <t1> <t2>
[aykizhvatkina@vbox lab14]$ ./1.sh 10 10 &>/dev/pts/1
[aykizhvatkina@vbox lab14]$ ./1.sh 10 10 &>/dev/pts/1

[aykizhvatkina@vbox lab14]$ sudo ./1.sh 10 10
Ресурс освобожден, начало использования на 10 секунд
Ресурс занят, ожидание освобождения...
Ресурс освобожден
, использование завершено
[aykizhvatkina@vbox lab14]$ Ресурс занят, ожидание освобождения...
```

Рис. 2.5: Запуск программы

Создаем файл 2.sh и устанавливаем право на выполнение. (рис. 2.6)

```
[aykizhvatkina@vbox lab14]$ touch 2.sh
[aykizhvatkina@vbox lab14]$ chmod +x 2.sh
```

Рис. 2.6: Создание файла

Реализовываем 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. (рис. 2.7 и рис. 2.8)


```

2.sh      [B---] 0 L:[ 1+ 0 1/ 16] *(0 / 375b) 00[*][X]
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Использование: $0 <название_команды>"
    exit 1
fi

command_name=$1

man_directory="/usr/share/man/man1"

if [ -f "$man_directory/$command_name.1.gz" ]; then
    zcat "$man_directory/$command_name.1.gz" | less
else
    echo "Справка для команды '$command_name' не найдена"
fi

```

Рис. 2.7: Программа

Проверяем выполнение программы. (рис. 2.8)

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "September 2024" "GNU coreutils 9.5" "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 specified.
.PP
Mandatory arguments to long options are mandatory for short options to
o.
.TP
\fB\-\a\fR, \fB\-\all\fR
do not ignore entries starting with .

```

Рис. 2.8: Запуск программы

Создаем файл 3.sh и устанавливаем право на выполнение. (рис. 2.9)

```
[aykizhvatkina@vbox lab14]$ touch 3.sh
[aykizhvatkina@vbox lab14]$ chmod +x 3.sh
```

Рис. 2.9: Создание файла

Используя встроенную переменную \$RANDOM, пишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. 2.10)

```
3.sh [BM--] 3 L: [ 1+ 0 1/ 17] *(3 / 463b) 0098 0x062
#!/bin/bash

generate_random_letter() {
    # Случайное число от 0 до 25
    random_number=$((RANDOM % 26))

    letter=$(printf "\\$(printf '%03o' $((65 + random_number)))")

    echo -n "$letter"
}

random_sequence=""
for ((i=0; i<10; i++)); do
    random_sequence="$random_sequence$(generate_random_letter)"
done

echo "Случайная последовательность букв латинского алфавита: $random_sequence"
```

Рис. 2.10: Программа

Проверяем выполнение программы. (рис. ??)

```
[aykizhvatkina@vbox lab14]$ ./3.sh
Случайная последовательность букв латинского алфавита: V)W)P)K)V)G)Z)W)N)T)
[aykizhvatkina@vbox lab14]$ ./3.sh
Случайная последовательность букв латинского алфавита: X)J)A)C)A)D)F)T)I)R)
```

{#fig:011width=70%

3 Выводы

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