

Отчет о выполнении лабораторной работы

Лабораторная работа №14

Филиппьева Ксения Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13

Список иллюстраций

3.1	семафоры	7
3.2	код в файле	8
3.3	активация кода	8
3.4	работоспособность кода	9
3.5	не баг, а фича	9
3.6	создание файла	9
3.7	код в файле	10
3.8	работоспособность кода	10
3.9	вывод кода	11
3.10	создание файла	11
3.11	код в файле	12
3.12	работоспособность кода	12

Список таблиц

1 Цель работы

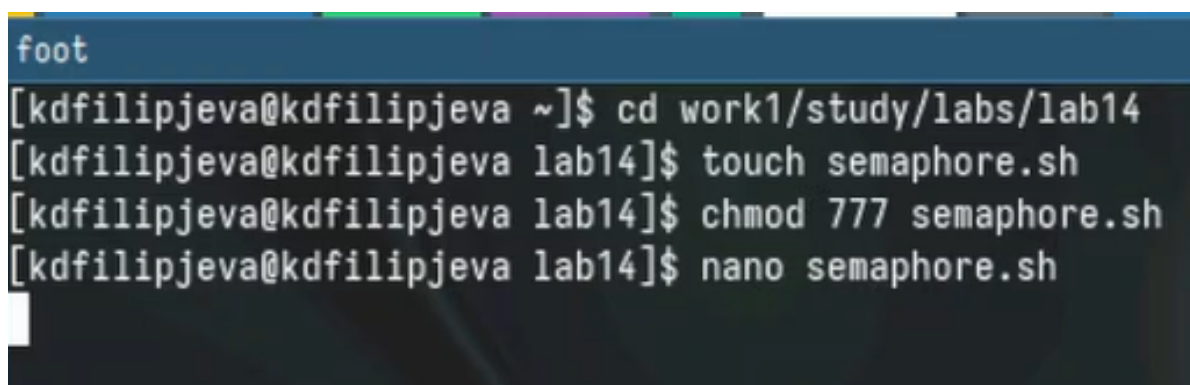
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 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

Создадим файл для первого задания, выдадим ему все права и войдем в него, чтобы вписать необходимый код (рис. 3.1).

A terminal window with a dark background and a blue title bar labeled 'foot'. The terminal shows a series of commands being executed by a user named 'kdfilipjeva' in a directory named 'lab14'. The commands are: 'cd work1/study/labs/lab14', 'touch semaphore.sh', 'chmod 777 semaphore.sh', and 'nano semaphore.sh'. The prompt for the last command is followed by a cursor.

```
foot
[kdfilipjeva@kdfilipjeva ~]$ cd work1/study/labs/lab14
[kdfilipjeva@kdfilipjeva lab14]$ touch semaphore.sh
[kdfilipjeva@kdfilipjeva lab14]$ chmod 777 semaphore.sh
[kdfilipjeva@kdfilipjeva lab14]$ nano semaphore.sh
```

Рис. 3.1: семафоры

Код для первого задания (рис. 3.2).

```
foot
GNU nano 7.2 semaphore.sh Изменён
#!/bin/bash

# Имя файла-семафора
SEMAPHORE_FILE="/tmp/semaphore"

# Функция для ожидания освобождения ресурса
wait_for_resource() {
    while [ -f "$SEMAPHORE_FILE" ]; do
        echo "Процесс $$ ожидает освобождения ресурса.."
        sleep 1
    done
}

# Функция для захвата ресурса
acquire_resource() {
    touch "$SEMAPHORE_FILE"
}
```

Рис. 3.2: код в файле

Активация кода и захват процесса (рис. 3.3).

```
[kdfilipjeva@kdfilipjeva lab14]$ ./semaphore.sh > /dev/t
ty2 &
[1] 24327
[kdfilipjeva@kdfilipjeva lab14]$
```

Рис. 3.3: активация кода

Активация кода во второй консоли, где он захватывается повторно, используется и высвобождается (рис. 3.4).


```
[kdfilipjeva@kdfilipjeva lab14]$ sudo ./semaphore.sh
Процесс 24513 захватил ресурс.
Процесс 24513 использует ресурс в течение
3 секунд...
Процесс 24513 освободил ресурс.
[kdfilipjeva@kdfilipjeva lab14]$
```

Рис. 3.4: работоспособность кода

Автоматическое завершение процесса при очистке терминала (рис. 3.5).

```
foot
[1]+  Завершён      ./semaphore.sh > /dev/tty2
[kdfilipjeva@kdfilipjeva lab14]$
```

Рис. 3.5: не баг, а фича

Создадим файл для второго задания и выдадим ему права (рис. 3.6).

```
[kdfilipjeva@kdfilipjeva lab14]$ touch myman.sh
[kdfilipjeva@kdfilipjeva lab14]$ chmod 777 myman.sh
[kdfilipjeva@kdfilipjeva lab14]$ nano myman.sh
```

Рис. 3.6: создание файла

Впишем необходимый код (рис. 3.7).

```
foot
GNU nano 7.2      myman.sh      Изменён
#!/bin/bash

# Проверяем, передано ли название команды в качестве>
if [ $# -eq 0 ]; then
    echo "Использование: $0 <команда>"
    exit 1
fi

# Путь к каталогу с архивами справочных файлов
man_dir="/usr/share/man/man1"
```

Рис. 3.7: код в файле

Активируем код для команды *ls* (рис. 3.8).

```
[kdfilipjeva@kdfilipjeva lab14]$ nano myman.sh
[kdfilipjeva@kdfilipjeva lab14]$ ./myman.sh ls
```

Рис. 3.8: работоспособность кода

Вывод активированного кода для команды *ls* (рис. 3.9).

```
foot
ESC[4mLSESC[24m(1) User Commands
ESC[4mLSESC[24m(1)
ESC[1mNAMEESC[0m
ls - list directory contents
ESC[1mSYNOPSISESC[0m
ESC[1mls ESC[22m[ESC[4mOPTIONESC[24m]... [ESC[
4mFILEESC[24m]...
```

Рис. 3.9: вывод кода

Создадим файл для третьего задания (рис. 3.10).

```
foot
[kdfilipjeva@kdfilipjeva lab14]$ touch random.sh
[kdfilipjeva@kdfilipjeva lab14]$ chmod 777 r
random.sh report/
[kdfilipjeva@kdfilipjeva lab14]$ chmod 777 random.sh
[kdfilipjeva@kdfilipjeva lab14]$ nano random.sh
```

Рис. 3.10: создание файла

Впишем необходимый нам код (рис. 3.11).

```
foot
GNU nano 7.2      random.sh      Изменён
#!/bin/bash

# Функция для генерации случайной буквы
generate_random_letter() {
    local ascii_code=$((65 + RANDOM % 26))
    printf "\\$(printf '%03o' $ascii_code)"
}

# Запрашиваем у пользователя длину последовательности
read -p "Введите длину последовательности: " length
```

Рис. 3.11: код в файле

Проверка кода (рис. 3.12).

```
[kdfilipjeva@kdfilipjeva lab14]$ ./random.sh
Введите длину последовательности: 15
Сгенерированная последовательность: QRSZGNOYMXVPPWA
[kdfilipjeva@kdfilipjeva lab14]$
```

Рис. 3.12: работоспособность кода

4 Выводы

Мы получили новые и отработали уже имеющиеся навыки программирования в оболочке ОС Linux.