## **РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

### **Факультет физико-математических и естественных наук**

### **Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 12**

дисциплина: Операционные системы

**Студент:**

Афтаева Ксения Васильевна

**Преподаватель:**

Велиева Т.В.

**Группа:** НПИбд-01-20

**МОСКВА** 2021 г.

### **Цель работы**:

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

### **Задачи**:

1. Изучить теоретический материал
2. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

* -iinputfile — прочитать данные из указанного файла;
* -ooutputfile — вывести данные в указанный файл;
* -pшаблон — указать шаблон для поиска;
* -C — различать большие и малые буквы;
* -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые *ключом -p*.

1. Написать на языке **Си** программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции *exit(n)*, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
2. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно **от 1 до N** (*например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.*). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).
3. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).
4. Ответить на контрольные вопросы

### **Объект и предмет исследования**:

Программирование в оболочке **OC UNIX/Linux**

### **Техническое оснащение**:

Ноутбук, на котором установлена виртуальная машина с линукс

### **Теоретические вводные данные** [[1]](https://eternalhost.net/blog/sistemnoe-administrirovanie/linux-bash-chto-eto) :

**bash** (*сокр. от «Bourne-Again shell»*) — это командная оболочка (или «интерпретатор командной строки»), используемая по умолчанию в операционных системах на базе Unix и Linux, созданная в 1989 году Брайаном Фоксом с целью усовершенствования командной оболочки sh.

**bash** позволяет автоматизировать различные задачи, устанавливать программное обеспечение, настраивать конфигурации для своего рабочего окружения и многое другое.

Основные преимущества:

* Позволяет работать со структурами «[[» (в sh доступна только «[» с ограничениями)
* Поддерживает работу с массивами в Линуксе
* Доступно множество расширений, выполненных по стандартам C, включая циклы с тремя аргументами «for((i=0;i<=3;i++))», возможность присваивать инкремент «+=» и многое другое
* Поддерживает синтаксис «<<<‘here strings’»
* Работает с расширениями «.{png,jpg}»
* Доступны алиасы для перенаправления, подобно «Csh», подобно «&|» для «2>&1 |» и «&>» для «> … 2>&1»
* Поддерживает сопроцессы с перенаправлением «<>»
* Огромный комплект расширений нестандартных конфигураций, включая изменение регистра
* Существенно увеличены возможности арифметики (правда, нет поддержки чисел с плавающей точкой)
* Переменные «SECONDS», «$PIPESTATUS[@]» и «$FUNCNAME» в Bash являются расширениями
* Доступно огромное количества функций, обеспечивающих работу в интерактивном режиме. Хотя на поведение скриптов они не влияют

**Циклы BASH**

Циклы позволяют выполнять один и тот же участок кода необходимое количество раз. В большинстве языков программирования существует несколько типов циклов. Большинство из них поддерживаются оболочкой Bash.

* for - позволяет перебрать все элементы из массива или использует переменную-счетчик для определения количества повторений;
* while - цикл выполняется пока условие истинно;
* until - цикл выполняется пока условие ложно.

Bash позволяет использовать циклы как в скриптах, так и непосредственно в командной оболочке.

### **Условные обозначения и символы**:

– $\* — отображается вся командная строка или параметры оболочки;

– $? — код завершения последней выполненной команды;

– $$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;

– $! — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;

– $- — значение флагов командного процессора;

– ${#\*} — возвращает целое число — количество слов, которые были результатом $\*;

– ${#name} — возвращает целое значение длины строки в переменной name;

– ${name[n]} — обращение к n-му элементу массива;

– ${name[\*]} — перечисляет все элементы массива, разделённые пробелом;

– ${name[@]} — то же самое, но позволяет учитывать символы пробелы в самих переменных;

– ${name:-value} — если значение переменной name не определено, то оно будет заменено на указанное value;

– ${name:value} — проверяется факт существования переменной;

– ${name=value} — если name не определено, то ему присваивается значение value;

– ${name?value} — останавливает выполнение, если имя переменной не определено, и выводит value как сообщение об ошибке;

– {name-value}. Если переменная определена, то подставляется value;

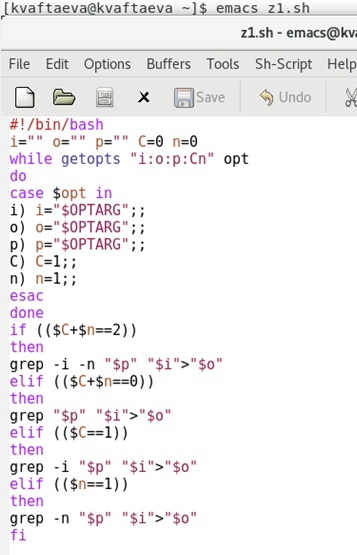
– ${name#pattern} — представляет значение переменной name с удалённым самым коротким левым образцом (pattern);

– ${#name[\*]} и ${#name[@]} — эти выражения возвращают количество элементов в массиве name.

### **Выполнение работы**:

**Задание 1**

1. Создаю файл для выполнения данного задания и сразу открываю его с помощью команды emacs z1.sh, где **z1.sh** название файла. Написала программу, выполняющую требуемые функции (Рис.1)

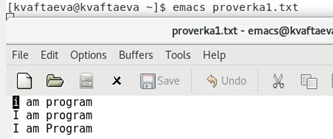


*Рис.1 Программа к заданию 1*

Её структура: В первой строке мы вызываем интерпретатор, в нашем случае **bash**. Далее идет блок объявления нужных перемнных, которые изначально пусты или равны нулю. Затем, используя пример применения оператора **getopts** из материалов к лабораторной работе №11, а также циклы **if** и **elif**, которые будут помогать распознать, какие именно действия нам нужно выполнить в зависимости от упоминания ключей *-C* и *-n*. Сами действия выполняются в строках **grep**.

После написания программы, созраняю ее комбинацией клавиш C-x C-s и закрываю редактор.

1. Командой emacs proverka1.txt, где **proverka1.txt** - название файла, создаю файл для проверки работы программы. Записываю в этот файл строки, отличающиеся только заглавными буквами (Рис.2)



*Рис.2 Файл для проверки 1 задания*

После чего сохраняю его комбинацией клавиш C-x C-s

1. Вызываю командный файл на выполнение, строкой ./z1.sh -i proverka1.txt -o zproverka.txt -p program -C -n. Здесь **z1.sh** - название командного файла, **proverka1.txt** - название файла, с которого мы будем считывать, **zproverka.txt** - название файла, в который мы будем записывать (создается автоматически), **program** - слово для поиска. Обозначаем сразу две опции. С помощью команды cat zproverka.txt просматриваю файл **zproverka.txt**, чтобы убедиться в правильности выполнения (Рис.3)

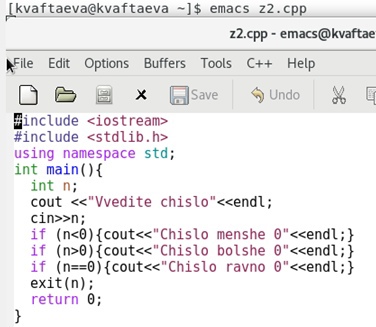


*Рис.3 Выполнение программ 1 задания*

Видим, что все выполнено верно

**Задание 2**

1. Создаю файл для выполнения данного задания и сразу открываю его с помощью команды emacs z2.cpp, где **z2.cpp** название файла. Написала программу на языке программирования С++, выполняющую нужные действия (Рис.4)

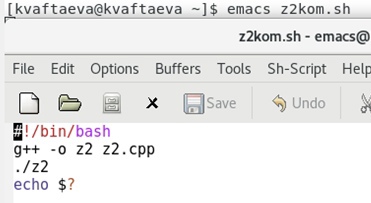


*Рис.4 Программа к заданию 2*

Ее структура: в первых трех строках я подключаю необходимые библиотеки и и инициализирую пространство имен. Далее приступаем к основной части программы. Инициализируем переменную **n** для хранения числа (**int n)**. Считываем значение переменной с клавиатуры (**cin>>n**). Далее мы с помощью **if** и разных условий (>,<,=) сравниваем число с нулем и выводим (**cout**) соответсвующую надпись. Затем программа завершается с помощью функции **exit(n)**.

После написания программы, сохраняю ее комбинацией клавиш C-x C-s и закрываю текстовый редактор.

1. Создаю и открываю командный файл для 2 задания, командой emacs z2kom.sh, где **z2kom.sh** - название этого файла. Написала командный файл (Рис.5)

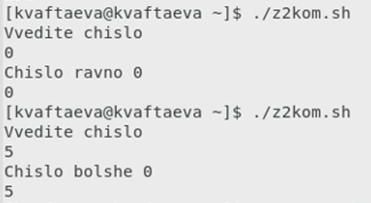


*Рис.5 Командный файл к заданию 2*

Его структура: в первой строке мы вызываем интерпретатор, в нашем случае **bash**. Во второй строке мы компилируем файл, а в третьей вызываем на выполнение (**./z2**). В конце анализируем и передаем на экран с помощью echo $?, какое число было введено для сравнения с нулем.

После написания сохрняю файл комбинцией клавиш C-x C-s и закрываю текстовый редактор.

1. Запускаю командный файл командой ./z2kom.sh. У нас появляется запрос на ввод числа, куда я ввожу несколько цифр для проверки (для каждого числа запускаю программу заново) (Рис.6)

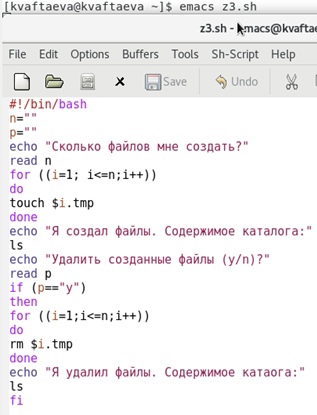


*Рис.6 Выполнение программы задание 2*

Видим, что программа выводит нужную фразу и число, т.е работает верно

**Задание 3**

1. Создаю и открываю командный файл для 3 задания, командой emacs z3.sh, где **z3.sh** - название этого файла. Написала командный файл (Рис.7)

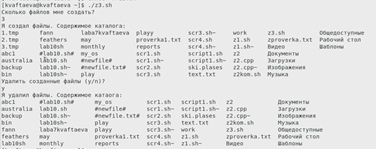


*Рис.7 Командный файл к заданию 3*

Его структура: в первой строке мы вызываем интерпретатор, в нашем случае **bash**. Далее инициализируем переменные для хранения количества файлов (**n**) и считывания ответа (**p**). Далее идет строка, спрашивающая у пользователя сколько файлов создать и считывает это количество с клавиатуры (**read n**). Далее циклом, выполняяющимся n раз, мы создаем файлы с нужным названием. После этого программа выводит содержимое каталога, чтобы можно было увидеть, что файлы созданы (команда ls). Далее программа спрашивает, нужно ли удалить файлы и считывает ответ с клавиатуры. Если ответ **y**, то циклом мы удаляем все файлы. После чего снова выводится содержимое каталога, чтобы можно было увидеть, что файлы удалены (ls)

После написания сохрняю файл комбинцией клавиш C-x C-s и закрываю текстовый редактор.

1. Запускаю командный файл командой ./z3.sh. У нас появляется запрос на ввод количества файлов, которые нужно создать. После чего эти файлы создаются. Далее мы видим запрос на удаление. Выбираем удалить. (Рис.8)

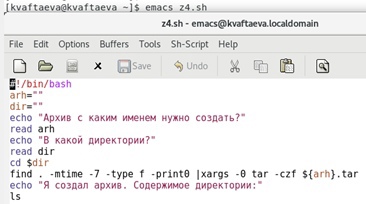


*Рис.8 Выполнение программы задание 3*

Видим по содержимому каталога, которое программа выводит дважды, что сначала файлы были созданы, а потом удалены

**Задание 4**

1. Создаю и открываю командный файл для 4 задания, командой emacs z4.sh, где **z4.sh** - название этого файла. Написала командный файл (Рис.9)

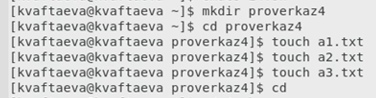


*Рис.9 Командный файл к заданию 4*

Его структура: в первой строке мы вызываем интерпретатор, в нашем случае **bash**. Далее мы инициализируем переменные для имени директории и имени архива. Запрашиваем имя архива, который будем создаавать, и имя директории, в которой будем работать. И вводим их с клавиатуры (read). переходим в нужный каталог (**cd $dir**). Далее мы начинаем поиск **find**. В этой строке: . - поиск осуществляется в текущем каталоге, -mtime -7 - файлы, редактированные не позднее чем 7 дней назад, -type f - поиск именно файлов, -print0 - позволяет выводить полный путь к файлу на стандартном выходе, за которым следует нулевой символ. Далее используем конвейер и создаем архив с заданным с клавиатуры именем при помощи команды tar, xarg - флаг -0 xargs используем, чтобы поместить все найденные файлы в архив. Ключи -czf помогут создать архив в linux, сжать архив с помощью Gzip, обозначить файлы для записи архива. Поссле чего выводим содержимое каталога, чтобы убедиться в том, что все выполнено

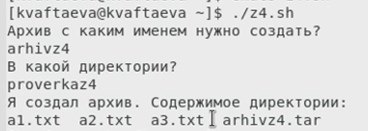
После написания сохрняю файл комбинцией клавиш C-x C-s и закрываю текстовый редактор.

1. С помощью команды mkdir создаю каталог **proverkaz4**, перехожу в него с помощью команды cd. Далее создаю в нем файлы **a1.txt**, **a2.txt**, **a3.txt** с помощью команды touch. После этого возвращаюсь в домашний каталог (Рис.10)



*Рис.10 Создание папки и файлов для проверки*

1. Запускаю командный файл командой ./z4.sh. Выводится запрос на ввод имени директории (я буду рботь в той, что создала в пункте №2) и имени архива (называю **arhivz4**) (Рис.11)



*Рис.11 Выполнение программы задание 4*

По выведенному содержимому каталога видим, что архив был создан

### **Контрольные вопросы**: [[2]](https://esystem.rudn.ru/pluginfile.php/1142377/mod_resource/content/2/008-lab_shell_prog_1.pdf)

1. Она осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.
2. При генерации имен файлов используют метасимволы:

"\*" - произвольная (возможно пустая) последовательность символов;

"?" - один произвольный символ;

"[...]" - любой из символов, указанных в скобках перечислением и/или с указанием диапазона;

"cat f\*" - выдаст все файлы каталога, начинающиеся с "f";

"cat *f*" - выдаст все файлы, содержащие "f";

"cat program.?" выдаст файлы данного каталога с однобуквенными расширениями, скажем "program.c" и "program.o", но не выдаст "program.com";

"cat [a-d]*" выдаст файлы, которые начинаются с "a", "b", "c", "d". Аналогичный эффект дадут и команды "cat [abcd]*" и "cat [bdac]\*".

1. for, case, if, while
2. Break, continue
3. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда test, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
4. Означает условие существования файла mani.$s
5. Если речь идет о 2-х параллельных действиях, то это while. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем until.

### **Заключение**:

Теоретический материал изучен и пригодится в дальнейшей работе. Все цели и задачи выполнены.

### **Вывод**:

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

### **Библиографический список**:

[1]: [Программирование в bash](https://eternalhost.net/blog/sistemnoe-administrirovanie/linux-bash-chto-eto)

[2]: [Описание лабораторной работы №11](https://esystem.rudn.ru/pluginfile.php/1142377/mod_resource/content/2/008-lab_shell_prog_1.pdf)