Лабораторная работа №1. Основы использования консольного интерфейса ОС Linux.
Рассматриваемые вопросы:

1. Работа с документацией по командам интерпретатора
2. Использование консольного текстового редактора
3. Создание скриптов для интерпретатора bash

Для получения подробного **справочного руководства** по любой команде можно набрать в консоли «man название команды», для кратной справки — название_команды -h или название_команды --help. Примеры: *man man* – справочное руководство по команде man; *man bash* – справочное руководство по интерпретатору bash.

Shell-скрипт – это обычный текстовый файл, в который последовательно записаны команды, которые пользователь может обычно вводить в командной строке. Файл выполняется командным интерпретатором – шеллом (shell). В Linux- и Unix-системах для того, чтобы бинарный файл или скрипт смогли быть запущены на выполнение, для пользователя, который запускает файл, должны быть установлены соответствующие права на выполнение. Это можно сделать с помощью команды `chmod u+x имя_скрипта`. В первой строке скрипта указывается путь к интерпретатору `#!/bin/bash`.
Для создания скрипта можно воспользоваться текстовым редактором `nano` или `vi`, набрав имя редактора в командной строке.
Справочная система Linux основана на "мануалах". Для получения справки по той или иной команде нужно набрать `man имя_команды`. Например, команда `man bash` выдаст вам дополнительную информацию по синтаксису скриптов на языке bash и возможностях командного интерпретатора.
Ниже приводятся основные правила программирования на языке bash.
Комментарии. Строки, начинающиеся с символа # (за исключением комбинации `#!`), являются комментариями. Комментарии могут также располагаться и в конце строки с исполняемым кодом.
Особенности работы со строками. Одиночные кавычки (`' '`), ограничивающие строки с обеих сторон, служат для предотвращения интерпретации специальных символов, которые могут находиться в строке. Двойные кавычки (`" "`) предотвращают интерпретацию специальных символов, за исключением `$, ` (обратная кавычка) и \ (escape – обратный слэш). Желательно использовать двойные кавычки при обращении к переменным. При необходимости вывести специальный символ можно также использовать экранирование: символ \ предотвращает интерпретацию следующего за ним символа.
Переменные. Имя переменной аналогично традиционному представлению об идентификаторе, т.е. именем может быть последовательность букв, цифр и подчеркиваний, начинающаяся с буквы или подчеркивания. Когда интерпретатор встречает в тексте сценария имя переменной, то он вместо него подставляет значение этой переменной. Поэтому ссылки на переменные называются подстановкой переменных. Если `variable1` – это имя переменной, то `$variable1` – это ссылка на ее значение. "Чистые" имена переменных, без префикса `$`, могут использоваться только при объявлении переменной или при присваивании переменной некоторого значения. В отличие от большинства других языков программирования, Bash не производит разделения переменных по типам. По сути, переменные Bash являются строковыми переменными, но, в зависимости от контекста, Bash допускает целочисленную арифметику с переменными. Определяющим фактором здесь служит содержимое переменных.
Оператор присваивания "=". При использовании оператора присваивания нельзя ставить пробелы слева и справа от знака равенства. Если в процессе присваивания требуется выполнить арифметические операции, то перед записью арифметического выражения используют оператор `let`, например:

```
Let a=2\*2
```

(оператор умножения является специальным символом и должен быть экранирован).
Арифметические операторы:

"+" сложение
"–" вычитание
"*" умножение
"/" деление (целочисленное)
"**" возведение в степень
"%" остаток от деления

Специальные переменные. Для Bash существует ряд зарезервированных имен переменных, которые хранят определенные значения.

– Позиционные параметры. Аргументы, передаваемые скрипту из командной строки, хранятся в зарезервированных переменных `$0, $1, $2, $3...`, где `$0` – это название файла сценария, `$1` – это первый аргумент, `$2` – второй, `$3` – третий и так далее. Аргументы, следующие за `$9`, должны заключаться в фигурные скобки, например: `${10}, ${11}, ${12}`. Передача параметров скрипту происходит в виде перечисления этих параметров после имени скрипта через пробел в момент его запуска.

– Другие зарезервированные переменные:
  `$DIRSTACK` – содержимое вершины стека каталогов
  `$EUID` – эффективный UID.
  `$UID` – ... содержит реальный идентификатор, который устанавливается только при логине.
  `$GROUPS` – массив групп к которым принадлежит текущий пользователь
  `$HOME` – домашний каталог пользователя
  `$HOSTNAME` – hostname компьютера
  `$HOSTTYPE` – архитектура машины.
  `$PWD` – рабочий каталог
  `$OSTYPE` – тип ОС
  `$PATH` – путь поиска программ
  `$PPID` – идентификатор родительского процесса
  `$SECONDS` – время работы скрипта (в секундах)
  `$#` – общее количество параметров, переданных скрипту
  `$*` – все аргументы, переданные скрипту (выводятся в строку)
  `$@` – то же самое, что и предыдущий, но параметры выводятся в столбик
  `$!` – PID последнего запущенного в фоне процесса
  `$$` – PID самого скрипта

Код завершения. Команда `exit` может использоваться для завершения работы сценария, точно так же как и в программах на языке С. Кроме того, она может возвращать некоторое значение, которое может быть проанализировано вызывающим процессом. Команде `exit` можно явно указать код возврата, в виде `exit nnn`, где `nnn` – это код возврата (число в диапазоне 0– 255).

Оператор вывода. `Echo переменные_или_строки`

Оператор ввода. `Read имя_переменной.` Одна команда `read` может прочитать (присвоить) значения сразу для нескольких переменных. Если переменных в `read` больше, чем их введено (через пробелы), оставшимся присваивается пустая строка. Если передаваемых значений больше, чем переменных в команде `read`, то лишние игнорируются.

Условный оператор. `If команда; then команда; [else команда]; fi.`

Если команда вернула после выполнения значение "истина", то выполняется команда после `then`. Если есть необходимость сравнивать значения переменных и/или констант, после `if` используется специальная команда `[[ выражение ]].` Обязательно ставить пробелы между выражением и скобками, например:

```
if [[ "$a" -eq "$b" ]]
then echo "a = b"
fi
```

Операции сравнения (в скобках указаны альтернативные формы записи):

`-z` # строка пуста
`-n` # строка не пуста
`=, (==)` # строки равны
`!=` # строки не равны
`-eq` # равно
`-ne` # не равно
`-lt, (<)` # меньше
`-le, (<=)` # меньше или равно
`-gt, (>)` # больше
`-ge, (>=)` # больше или равно
`!` # отрицание логического выражения
`-a, (&&)` # логическое «И»
`-o, (||)` # логическое «ИЛИ»

Множественный выбор. Для множественного выбора может применяться оператор `case`.

```
case переменная in
значение1 )
команда 1
 ;;
значение2 )
команда 2
 ;;
esac
```

Выбираемые значения обозначаются правой скобкой в конце значения. Разделитель ситуаций – `;;`

Цикл for. Существует два способа задания цикла for.

1. Стандартный – `for переменная in список_значений do команды done`. Например:

```
for i in 0 1 2 3
do
echo $i
done
```

2. С-подобный

```
for ((i=0; c <=3; i++))
do
echo $i
done
```

Цикл while: `while условие do команда done`. Синтаксис записи условия такой же, как и в условном операторе.

Управление циклами. Для управления ходом выполнения цикла служат команды `break` и `continue`. Они точно соответствуют своим аналогам в других языках программирования. Команда `break` прерывает исполнение цикла, в то время как `continue` передает управление в начало цикла, минуя все последующие команды в теле цикла.

Задание на лабораторную работу

1. Создайте свой каталог в директории /home/user/ Все скрипты создавайте внутри этого каталога или его подкаталогов. (`mkdir lab1`)
2. Напишите скрипты, решающие следующие задачи:
   i) В параметрах скрипта передаются две строки. Вывести сообщение о равенстве или неравенстве переданных строк.
   ii) В параметрах при запуске скрипта передаются три целых числа. Вывести максимальное из них.
   iii) Считывать строки с клавиатуры, пока не будет введена строка "q". После этого вывести последовательность считанных строк в виде одной строки.
   iv) Считывать с клавиатуры целые числа, пока не будет введено четное число. После этого вывести количество считанных чисел.
   v) Создать текстовое меню с четырьмя пунктами. При вводе пользователем номера пункта меню происходит запуск редактора nano, редактора vi, браузера links или выход из меню.
   vi) Если скрипт запущен из домашнего директория, вывести на экран путь к домашнему директорию и выйти с кодом 0. В противном случае вывести сообщение об ошибке и выйти с кодом 1.
3. Предъявите скрипты преподавателю и получите вопрос или задание для защиты лабораторной работы.
4. После защиты лабораторной работы удалите созданный директорий со всем его содержимым (`rm –R lab1`)