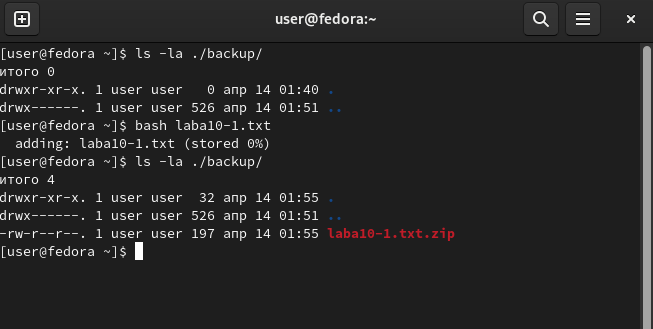
**10.3.1. Листинг программы (файл laba10-1.txt):**

*zip ~/backup/$0.zip $0*

**10.3.1. Результаты выполнения:**



**10.3.2. Листинг программы (файл laba10-2.txt):**

*for i*

*do*

*echo $i*

*done*

**10.3.2. Результаты выполнения:**



**10.3.3. Листинг программы (файл laba10-3.txt):**

*for x in echo $1/\**

*do*

*if [ -r "$x" ]*

*then*

*echo -n R*

*else*

*echo -n -*

*fi*

*if [ -w "$x" ]*

*then*

*echo -n W*

*else*

*echo -n -*

*fi*

*if [ -x "$x" ]*

*then*

*echo -n "X "*

*else*

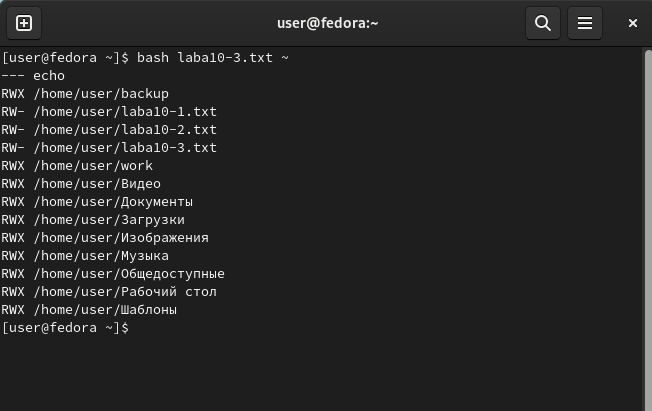
*echo -n "- "*

*fi*

*echo $x*

*done*

**10.3.3. Результаты выполнения:**



1. Командная оболочка- это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

– оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;

– BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

1. POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.
2. В языке программирования bash массивы и переменные определяются так: создание массива- используется команда set с флагом -A. За флагом следует имя переменной, а затем список значений, разделённых пробелами.
3. Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Команда let берет два операнда и присваивает их переменной. Положительным моментом команды let можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа let sum=x+7, и let будет искать переменную x и добавлять к ней 7. Команда let также расширяет другие выражения let, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда let не ограничена простыми арифметическими выражениями.показывает полный набор let-операций.

Команда **read** читает одну строку из стандартного входного потока и записывает ее содержимое в указанные переменные. Если задана единственная переменная, в нее записывается вся строка. В результате ввода команды **read** без параметров строка помешается в переменную среды $reply. При указании нескольких переменных в первую из них записывается первое слово строки, во вторую — второе слово и т. д. Последней переменной присваивается остаток строки.

1. Скриптовой **язык** **bash** умеет выполнять сложение, вычитание, умножение, целочисленное деление и получение остатка от деления.
2. Скриптовой **язык** **bash** умеет выполнять сложение, вычитание, умножение, целочисленное деление и получение остатка от деления.
3. Двойные скобочки применяются для арифметических выражений/
4. HOME — имя домашнего каталога пользователя. Если команда cd вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. – IFS — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line). – MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение You have mail (у Вас есть почта). – TERM — тип используемого терминала. – LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему. В командном процессоре Си имеется ещё несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды set.
5. Метасимволы — это **символы, имеющие в регулярном выражении специальное значение**.
6. Создать файл с любым именем. Написать в него любой код, но первую строчку сделать #!/bin/bash # означает комментарий, и оболочка по этой строке определит, как запускать командный файл.
7. **function** *function\_name* {  
   *command*...  
   }

или

*function\_name* () {  
*command*...  
}

1. 1Используйте цикл for в Bash и просматривайте все переданные аргументы, используя $@ argument.

2Внутри проверки цикла for, является ли предоставляемым аргументом каталог или нет, используется оператор if с использованием опции -d для первого аргумента с использованием параметра $ 1. Если это true, то выведите сообщение о том, что предоставленный аргумент является каталогом.

3Если аргумент не является каталогом, то проверьте его на наличие файла. Используйте параметр -f и оператор if для первого аргумента, используя параметр $1. Если это true, то выведите сообщение, в котором указан аргумент в файле.

4Если оба условия ложны, то ясно, что предоставленный аргумент не является ни файлом, ни каталогом. Затем выведите сообщение, что данный аргумент не является ни файлом, ни каталогом.

1. Команда **set** изменяет значения внутренних переменных сценария. Команда **unset** удаляет переменную, фактически -- устанавливает ее значение в null.  [typeset](https://www.opennet.ru/docs/RUS/bash_scripting_guide/x4704.html) задаёт и/или накладывает ограничения на переменные
2. Мы можем использовать программу / команду getopts для анализа аргументов, переданных скрипту в командной строке / терминале, с помощью циклов и операторов переключения.
3. – $\* — 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} — это выражение работает противоположно ${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