Лабораторная работа №1. Основы использования shell скриптов.

Задание на лабораторную работу:

1. Выберите shell-интерпретатор и изучите его документацию (рекомендуется bash);
2. Используя язык интерпретатора ***sh** (bash, zsh, ksh, fish, и т.д.) разработайте приложение, соответствующее следующим требованиям:
    2.1. приложение принимает набор параметров запуска (ключей), определяющих его работу;
    2.2. в случае, если первым аргументом передан ключ **calc**, приложение выполняет функции калькулятора. Если вторым аргументом передан ключ **sum/sub/mul/div**, приложение выводит на экран сумму/разность/произведение/частное третьего и четвертого аргумента, являющихся целыми числами;
    2.3. в случае, если первым аргументом передан ключ **search**, приложение производит рекурсивный поиск по содержимому файлов в директории, указанной вторым аргументом, и выводит на экран строки в соответствии с регулярным выражением, заданным третьим аргументом;
    2.4. в случае, если первым аргументом передан ключ **reverse**, приложение в обратном порядке записывает содержимое файла, имя которого задано вторым аргументом, в файл с именем, переданным третьим аргументом;
    2.5. в случае, если первым аргументом передан ключ **strlen**, приложение должно вывести количество символов в строке, переданной вторым аргументом. Использовать команду **wc** и циклы запрещено;
    2.6. в случае, если первым аргументом передан ключ **log**, приложение должно вывести строки файла */var/log/anaconda/*X.*log*, содержащие предупреждения и информационные сообщения, заменив маркеры предупреждений и информационных сообщений на слова *Warning:* и *Information:*, чтобы в получившемся файле сначала шли все предупреждения, а потом все информационные сообщения. При выводе на экран слово *Warning* вывести желтым цветом, слово *Information* — голубым цветом;
    2.7. в случае, если первым аргументом передан ключ **exit**, приложение завершает свою работу с кодом возврата, заданным вторым параметром. Если код возврата не задан, по умолчанию используется 0;

2.8. в случае, если первым аргументом передан ключ **help**, приложение выводит справку по использованию, в которой перечислены все действия и их аргументы;

2.9. в случае, если первым аргументом передан ключ **interactive**, приложение переходит в интерактивный режим работы, предоставляя пользователю интерактивное меню с выбором действий:

2.9.1. пункты меню обозначены буквами, выбор действия осуществляется вводом соответствующей буквы или названия действия;

2.9.2. после выбора действия пользователю предлагается ввести аргументы; допускается реализация как единого приглашения для ввода всех аргументов в строку, так и запрос каждого аргумента команды по отдельности;

2.9.3. после выполнения действия, приложение возвращается в меню;

2.9.4. пункты меню соответствуют действиям не интерактивного режима (calc, search, reverse, exit и т.д.);

2.10. приложение должно быть разделено на несколько взаимодействующих модулей (скриптов), соответственно выполняемым ими задачам;

2.11. при разработке приложения необходимо руководствоваться здравым смыслом, принципами единой ответственности и переиспользования кода;

2.12. приложение должно корректно обрабатывать исключительные ситуации:

2.12.1. при вводе некорректного имени действия должно выводиться сообщение об ошибке и справка по использованию приложения;

2.12.2. при вводе некорректного значения аргумента должно выводиться сообщение об ошибке с объяснением причины;

2.12.3. при возникновении исключительных ситуаций при работе вызываемых программ, сообщения об ошибках, произошедших в процессе их работы, не должны выводиться пользователю; приложение должно обработать возникающие ошибки, и, если ошибка является критической, выдать пользователю сообщение об ошибке с описанием ее причины;

2.12.4. в случае, если ошибка произошла в интерактивном режиме, выдается сообщение об ошибке, как описано выше, и приложение возвращается в меню;

2.12.5. в случае, если нарушена целостность приложения, и отсутствует часть его скриптов, при запуске должно быть выдано предупреждение о том,

какие скрипты недоступны; если отсутствующие скрипты не являются критически важными для работы остальной части приложения, приложение продолжает работу;

2.12.6. в случае, если часть функций приложения недоступна из-за отсутствия модулей, допускаются следующие реализации:

2.12.6.1. При попытке вызова недоступного действия в неинтерактивном режиме пользователю выдается сообщение ошибке. При работе в интерактивном режиме пользователю не предлагается вызов недоступного действия. При загрузке приложения в интерактивном режиме выдается сообщение о недоступности действий и причинах этого;

2.12.6.2. При попытке вызова недоступного действия в неинтерактивном режиме пользователю выдается сообщение ошибке. При работе в интерактивном режиме пользователю предлагается выполнение действия, однако при попытке его вызова выдается сообщение об ошибке. При загрузке приложения в интерактивном режиме выдается сообщение о недоступности действий и причинах произошедшего.

2.12.7. при возникновении критической ситуации в приложении, оно должно выдавать сообщение об ошибке, и происходить завершение приложения с отрицательным кодом возврата;

2.12.7.1. значение и описание кодов возврата должно быть описано в **help**.

2.12.8. Все сообщения об ошибках должны выводиться в стандартный поток ошибок (stderr).

3. Опубликуйте результаты работы в публичном репозитории.

Рекомендации:

1. Структурируйте код скрипта: используйте функции, разбивайте приложение на отдельные файлы.
2. Используйте конвейеры.
3. Используйте сокращенную запись условий ([[$a -eq $b]] && doThen || doElse).
4. Пишите комментарии: вам проще разрабатывать, преподавателю проще проверять.
5. Не забывайте поэтапно выкладывать результаты работы на GitHub. В случае списывания с вашей работы, коммиты будут весомым доказательством самостоятельности выполнения работы;
6. Не забывайте писать шебанг (#!) в начале скриптов (напр, #!/bin/bash).