Семинар #7: Скрипты языка Bash. Практика.

Как сдавать задачи

Для сдачи ДЗ вам нужно создать репозиторий на GitLab (если он ещё не создан) под названием `devtools-homework`. Структура репозитория должна иметь вид:

```
├── seminar7_bash_scripting/
│   ├── 01.sh
│   ├── 02.sh
│   └── ...
└── ...
```

Для каждой задачи, если в самой задаче не сказано иное, нужно создать 1 скрипт с расширением `.sh` и шебангом в начале скрипта. Если задача делится на подзадачи нужно, если в самой задаче не сказано иное, создать скрипт для каждой подзадачи. Названия файлов решений для всех задач/подзадач должны начинаться с номера задачи, например `01.sh` или `04b.sh`, даже если в условии задачи используется другое имя для скрипта.

Переменные

Задача 1. Новая переменная

Пусть есть следующая простая программа на языке Bash:

```
#!/bin/bash
echo "Hello, my name is Alice."
echo "Alice, welcome to the world of bash scripts!"
echo "Today Alice will learn about bash scripting."
```

Это скрипт можно найти в `seminar7_bash_scripting/practice/hello_alice.sh`. Измените этот скрипт, добавив переменную `name` со значением `"Alice"` и три раза используйте эту переменную в командах `echo`.

Задача 2. Печатаем переменные среды

Напишите скрипт, который будет печатать следующие переменные среды, используя команду `echo`:

- `PATH` – переменная среды, которая содержит список каталогов, в которых оболочка ищет исполняемые файлы (команды).
- `USER` – переменная среды, которая содержит имя текущего пользователя.
- `UID` – переменная среды, которая содержит UID текущего пользователя.
- `HOME` – имя домашней директории текущего пользователя.
- `SHELL` – имя текущей используемой оболочки.
- `HOSTNAME` – имя компьютера на котором исполняется программа.
- `LANG` – язык и локаль для сообщений программ, например `en_US.UTF-8` или `ru_RU.UTF-8`.
- `PS1` – переменная, которая содержит строку перед вводом команды. Что будет, если присвоить этой переменной значение `"$ "`? Перезайдите в терминал, если хотите восстановить старое значение переменной.

Используйте команду `env`, чтобы напечатать все переменные среды и убедиться, что ваш скрипт печатает корректные значения.

Задача 3. Создаём свою переменную среды

Выполните последователь следующие команды в терминале. Для каждой команды, определите, что будет печататься на экран.

(a) Создайте новую переменную по имени `ALPACA` со значением `apple`.

```
$ ALPACA=apple
```

(b) Распечатайте значение переменной `ALPACA`:

```
$ echo "This is $ALPACA"
```

(c) Создайте новый файл `print_alpaca.sh`, который будет содержать следующий код:

```
#!/bin/bash
echo "This is $ALPACA"
```

После этого дайте этому файлу права на исполнение:

```
$ chmod +x print_alpaca.sh
```

(d) Определите, что напечатает следующая последовательность команд:

```
$ unset ALPACA
$ ALPACA=apple
$ ./print_alpaca.sh
```

Команда `unset` удаляет переменную `ALPACA`. После этого создаётся новая переменная со значением `apple`. Это делается для чистоты эксперимента.

(e) При исполнении скрипта в оболочке создаётся новый процесс и обычные переменные оболочки в этом процессе не видны. Но видны экспортированные переменные, создаваемые командой `export`. Определите, что напечатает следующая последовательность команд:

```
$ unset ALPACA
$ export ALPACA=apple
$ ./print_alpaca.sh
```

(f) Можно задать экспортированные переменные для одной команды, с помощью специального синтаксиса:

```
$ VAR1=value1 VAR2=value2 command
```

Определите, что напечатает следующая последовательность команд:

```
$ unset ALPACA
$ ALPACA=apple ./print_alpaca.sh
$ ALPACA=apple; ./print_alpaca.sh
$ ALPACA=apple && ./print_alpaca.sh
```

(g) Команда `source` нужна для того, чтобы выполнить команды из скрипта в текущей оболочке, не создавая новый процесс. Определите, что будет печать следующие команды:

```
$ unset ALPACA
$ ALPACA=apple
$ source ./print_alpaca.sh
```

(h) Создайте экспортируемую переменную и убедитесь, что она работает:

```
$ export ALPACA=apple
$ ./print_alpaca.sh
```

После этого закройте терминал и заново его откройте. Что теперь напечатают команды:

```
$ ./print_alpaca.sh
$ source ./print_alpaca.sh
```

(i) Добавьте создание переменной среды `ALPACA` в файл `~/.bashrc`. В этом файле нужно добавить строку:

```
export ALPACA=apple
```

Скрипт `.bashrc` выполняется в процессе текущей оболочки при каждом запуске терминала. После этого закройте терминал и заново его откройте. Что теперь напечатает команда:

```
$ ./print_alpaca.sh
```

(j) Распечатайте все переменные среды, с помощью команды `printenv` и убедитесь, что добавилась новая переменная среды `ALPACA`.

Для того, чтобы сдать эту задачу создайте файл `03.txt` в котором будут печататься выводы команд на каждом шаге в следующем формате:

```
# Subtask a
$ ALPACA=apple
# Subtask b
$ echo "This is $ALPACA"
This is apple
# Subtask c
...
```

Задача 4. Внутренняя или внешняя

Часть команд, которые используются в bash являются встроенными командами `bash` (например, команды `cd`, `echo` и другие), а некоторые команды являются отдельными программами (например, `ls`, `cp` и другие). Некоторые команды реализованы обоими способами, в этом случае при выполнении команды обычно выбирается встроенная версия. Чтобы понять какая команда является встроенной, а какая команда является отдельной программой, можно использовать команду `type -a`. Определите, как реализованы следующие команды:

- `cd`
- `cp`
- `pwd`
- `ls`
- `touch`
- `mkdir`
- `echo`
- `source`
- `type`
- `which`

Для решения этой задачи создайте файл `04.txt` в котором будут представлены ответы в следующем формате:

```
cd - shell builtin
cp - programm /usr/bin/cp
...
```

Если команда реализована несколькими способами, то выбирайте первый – наиболее приоритетный.

Задача 5. Создаём новую команду

(a) Когда вы запускаете команду, если она не является встроенной, bash ищет эту программу в стандартных путях, задаваемых переменной `PATH`. Переменная `PATH` является строкой, в которой перечислены директории, разделённые символом `:`. Очистите переменную `PATH`, присвоив ей пустую строку.

```
$ export PATH=""
```

убедитесь, что после этого внешние команды перестали работать. **Перезайдите в терминал, чтобы восстановить значение переменной** `PATH`.

(b) Создайте простой скрипт `hello.sh`, который будет содержать следующий код:

```
#!/bin/bash
echo "Hello, Bash"
```

Дайте этому файлу права на исполнение и исполните его

```
$ chmod +x ./hello.sh
$ ./hello.sh
```

(c) Попробуйте исполнить этот файл, без указания текущей директории:

```
$ hello.sh
```

Без указания пути до файла bash будет искать такой файл в стандартных путях. Так как такого файла там не будет, данная команда не будет найдена.

(d) Переместите данный скрипт в один из стандартных путей из переменной `PATH`, например, в `/usr/local/bin`. Для этой операции понадобятся права суперпользователя. Убедитесь, что у скрипта есть право на исполнение для текущего пользователя, и, желательно, для всех пользователей. После этого запустите скрипт без указания директории:

```
$ hello.sh
```

Теперь этот файл должен найтись и скрипт запустится.

(e) Измените имя скрипта в системной директории с `hello.sh` на `hello`. Теперь скрипт можно будет запустить, как любую другую команду.

```
$ hello
```

Для того, чтобы сдать эту задачу создайте файл `05.txt` в котором нужно будет напечатать все команды, которые были исполнены при выполнении данного задания в следующем формате:

```
# Subtask a
$ export PATH=""
# Subtask b
...
```

Задача 6. Подмена ls

Аргументы

Задача 7. Привет скрипт

Напишите программу на языке bash, которая будет печатать имя своего файла. Например, если скрипт вызывается как `./script.sh`, то он должен печатать на экран строку `"Hello ./script.sh"`.

```
$ ./script.sh
Hello ./script.sh
$ mv script.sh 07.sh
$ ./07.sh
Hello ./07.sh
```

Задача 8. Количество аргументов

Напишите программу, которая будет печатать количество переданных ей аргументов.

```
$ ./script.sh
0
$ ./script.sh apple
1
$ ./script.sh apple banana cherry durian
4
```

Кавычки

Задача 9. Одинарные или двойные кавычки

Определите, что сделают следующие команды в bash:

(a)
```
touch alpha beta gamma
```

(b)
```
touch "alpha beta gamma"
```

(c)
```
touch 'alpha beta gamma'
```

(d)
```
echo /var/*
```

(e)
```
echo "/var/*"
```

(f)
```
echo '/var/*'
```

(g)
```
user=alice
echo hello $user
```

(h)
```
user=alice
echo "hello $user"
```

(i)
```
user=alice
echo 'hello $user'
```

(j)
```
echo I am in $(pwd) folder
```

(k)
```
echo "I am in $(pwd) folder"
```

(l)
```
echo 'I am in $(pwd) folder'
```

(m)
```
echo I have $100
echo I have \$100
```

(n)
```
echo "I have $100"
echo "I have \$100"
```

(o)
```
echo 'I have $100'
echo 'I have \$100'
```

(p)
```
touch hello.sh
find /usr -name *.sh
```

(q)
```
touch hello.sh
find /usr -name "*.sh"
```

Работа со строками

Задача 10. Базовое имя

В linux есть команда `basename`, которая принимает на вход полный путь до некоторого файла, а возвращает только имя этого файла.

```
$ basename /etc/passwd
passwd
```

```
$ basename /home/user/workspace/file.txt
file.txt
```

Вам нужно написать скрипт, который будет делать то же самое, но без использования программы `basename`.

Коды возврата

Задача 11. Коды возврата базовых программ

В каких случаях следующие программы возвращают ненулевой код возврата:

a. `ls`

b. `curl`

c. `diff`

d. `find`

e. `grep`

Ветвление

Задача 12. Существует ли файл

Напишите скрипт, который принимает название файл, через аргумент и печатает `Yes`, если такой файл (любого типа) существует в данной директории или `No`, если не существует. Решите эту задачу тремя способами:

(a) С использованием `if` и ключа `-e`.

(b) С использованием `if` и программы `ls`.

(c) С использованием операторов `&&`, `||` и программы `ls`.

Скрипт должен принимать ровно один аргумент. Если количество передаваемых аргументов меньше или больше одного, то программа должна завершаться с текстом:

```
Usage: $0 <filename>
```

Скрипт не должен выводить на экран никакого дополнительного текста, не указанного в условии задачи.

Задача 13. Удаление с сообщением

Пытается удалить файл, если его не существует, то должна напечатать `File not exists`.

(a) С использованием `if` и ключа `-f`.

(b) С использованием оператора `||` и программы `ls`.

Задача 14. Проверка пользователя

grep /etc/passwd.

(a) С использованием `if`.

(b) С использованием оператора `||`.

Задача 15. Бэкап всех файлов в текущей директории

Создать backup копию для каждого файла в этой директории в скрытой папке. Делает новый backup только есть файл изменился. Названия файлов `.backup/file.txt.0`, `.backup/file.txt.1`, и т.д.