Bash-scripts
Введение

BASH — Bourne-Again SHell
(Stephen Bourne – создатель sh),
GNU-реализация стандартного
интерпретатора команд Unix – систем.

Bash-scripts
Зачем нужен? (Пример)

- Не нужно много раз писать одно и тоже
- Сценарий очистки лог-файлов в /var/log:

cd /var/log
cat /dev/null > messages
cat /dev/null > wtmp

Bash-scripts
Что нужно знать с самого начала?

- Любой bash-скрипт должен начинаться со строки:
  #!/bin/bash
  (Последовательность #! называется *Sha-Bang* – указание интерпретатора)
- Запуск интерпретатора:
  bash scriptname [arguments]
- Комментарии начинаются с символа #

Bash-scripts
Запуск сценария

- *Bash scriptname args*
- Сделать исполняемым:
  chmod 555 scriptname
  а затем просто запустить:
  ./scriptname
- Если поместить сценарий в каталог */usr/local/bin*, то вызвать его можно просто набрав название файла в командной строке *(Видимо, не лучшая идея)*

Bash-scripts
Переменные

- В *BASH* переменные не имеют типа.
- Если variable -- это имя переменной, то $variable -- это ссылка на ее значение.
- Присваивание значений с помощью знака равенства (=) **!!!без пробелов с обеих сторон!!!**
  variable=*34*

Bash-scripts
Переменные(Пример)

```
a=375
b=$a
echo $b                                  #375
hello="A B    C      D"
echo hello                          # hello
echo $hello                         # A B C D
echo "$hello"                       # A B    C      D
echo '$hello'                       # $hello

var=23                               # Присваивание.
unset var                            # Сброс.
echo "var = $var"                    # var=
```

Bash-scripts
Массивы

- arr[0]=0 *# не нужна непрерывная*
  arr[5]=5 *последовательность*
- arr2=(ноль один два три четыре)
- arr3=([17]=семнадцать [21]=двадцать_один)
- **echo** ${arr[5]} # нужны { }

Bash-scripts
Переменные.Присваивание (простое)

- a=1 **!!!без пробелов с обеих сторон!!!**
- **let** a=16+5 *# **let** – арифм. действия*
- **for** a **in** 7 8 9 11
  do
  ...
  done
- **read** a

Bash-scripts
Присваивание (замаскированное)

- a=`echo Hello!`
  echo $a # Hello!
- a=`ls -l`
- a=$(ls) # Аналог предыдущей записи

Bash-scripts
Зарезервированные переменные

- $HOME - домашний каталог пользователя
- $OSTYPE - тип ОС
- $# - общее количество параметров переданных скрипту
- Все зарезервированные переменные можно посмотреть набрав ***set***

Bash-scripts
Позиционные параметры

- аргументы, передаваемые скрипту из командной строки -- $0, $1, $2, $3, ...
- $0 -- это название файла сценария
- $n -- это n-ный аргумент
- Аргументы, следующие за $9 – в фигурных скобках -- ${10}, ...
- $*, $@ -- специальные переменные, содержат все аргументы
- $# - количество переданных аргументов

Bash-scripts
команда test

- Команда **test** (или ”**[ ]**”)проверяет выполнение некоторого условия. С ее помощью формируются операторы выбора и цикла.
- Минус команды **test***:*

```
[ privet ]
echo $?                    #    0
[ ]
echo $?                    #    1
```

Test возвращает 0 (**истина**), если в скобках стоит непустое слово

Bash-scripts
команда test(продолжение)

Проверка файлов:

- -f file - файл "file" обычный файлом;
- -d file - файл "file" директория;

Сравнение чисел:

- x -eq y - "x" равно "y",
- x -ne y - "x" неравно "y",
- x -gt y - "x" больше "y",
- x -lt y - "x" меньше "y",

- Остальное – **man test**

Bash-scripts
Циклы

- **for**
  - **for** arg **in** list
    do
      command(s)
    done
  - **for** arg **in** $(seq n) # 1 2 3 ... n
  - **for** arg **in** {1..n} # 1 2 3 ... n
- **While** condition
  do
    command(s)
  done

Bash-scripts
Циклы(Пример)

```
#!/bin/bash
PSW=/etc/passwd
n=1                                # Число пользователей

for name in
        $(awk 'BEGIN{FS=":"}{print $1}' "$PSW" )
do
  echo "Пользователь #$n = $name"
  let "n += 1"
done

exit 0
```

Bash-scripts
Особенности.

- **for** arg **in** $*

Правильно:

- **for** arg **in** ”$@”
- **for** arg

Bash-scripts
Условные выражения

- **if** condition
  then
  command1
  else
  command2
  fi
- **if test** condition точно тоже, что и
  if [condition **]**
- **if [[** condition **]]** - возможны &&, ||
- **if ((** арифметическое выражение **))**

Bash-scripts
Условия. ошибки

- [[$foo > 7]] - сравнивает строки
- [$foo > 7] – перенаправление вывода

Правильно

- (($foo > 7)) - сравнивает числа
- [$foo -gt 7]
- [[$foo -gt 7]]

Bash-scripts
Условия. ошибки

- [bar == "$foo"] - неверно

Правильно

- [bar = "$foo"]
- [[bar == "$foo"]]

Bash-scripts

Условия. Множественный выбор. Пример.

```
echo "1 Запуск программы nano"
echo "2 Запуск программы vi"
echo "3 Выход"
read doing
case $doing in
1) /usr/bin/nano ;;
2) /usr/bin/vi ;;
3) exit 0;;
*) echo "Введено неправильное действие"
esac
```

Bash-scripts
Функции

- **function** function_name **{**
  command...
  }
- function_name () **{**
  command...
  }
  # не стоит скрещивать 2 определения!!!
- Нет опережающего объявления функции, НО...

Bash-scripts
Функции(продолжение)

```
foo1 (){
  echo "Вызов функции \"foo2\" из \"foo1\"."
foo2
}

foo2 (){
  echo "Функция \"foo2\"."
}

foo1
```

Bash-scripts
Функции(продолжение)

```
if [ "$USER" = student ]
 then
   student_greet ()   {
    echo "Привет, student!"
   }
 fi
 student_greet
```

 # Работает только у пользователя student, другие получат сообщение об ошибке.

Bash-scripts
Функции

- Функции могут принимать входные аргументы и возвращать код завершения.
- **return** – завершает исполнение функции. Может возвращать ”код завершения” (int), который записывается в переменную $?.
- Если ”код завершения” не указан – возвращается код последней команды в функции

Bash-scripts
Команда expr

- вычисляет заданное выражение
  !! аргументы должны отделяться пробелами!!

Примеры:

- **expr** 3 + 5 *#8*
- **expr** 5 * 3 *#15 экранируем **
- y=$(**expr** $y + 1) *#инкремент*

Bash-scripts
Команда bc

- **bc** -- утилита, выполняющая вычисления с произвольной точностью.
- **echo** "scale=4;(321-123)/123" | **bc -l**
  scale=4 – количество знаков после запятой
- **echo** "obase=16;ibase=10;123" | **bc**
  преобразование из десятичного в шестнадцатеричный вид
- var3=$(**bc -l** << EOF
  scale = 9; s (1.7)
  EOF)
  Использование со "встроенным документом"

Bash-scripts
Функции. Пример.

```
hyp=    # Объявление глобальной переменной.
hypotenuse ()    {
hyp=$(bc -l << EOF
scale = 9
sqrt ( $1 * $1 + $2 * $2 )
EOF
)
}
hypotenuse 3.68 7.31     # $hyp =  8.184039344
```

Bash-scripts
Операции со строками

- Длина строки *${#string}*
- Длина подстроки в строке
  expr "$string" : '$substring'

  stringZ=abcABC123ABCabc
  ***echo `expr** "$stringZ" : 'abc[A-Z]*.2'`* *# 8*
- *expr index $string $substring - № позиции совпадения в $string с символом в $substring.*
- Извлечение подстроки *${string:position}*
  либо *${string:position:length}*

Bash-scripts
Операции со строками

- *expr "$string" : '\($substring\)'*
  Находит и извлекает первое совпадение $substring в $string, где $substring -- это регулярное выражение.
- Удаление части строки
  ${string#substring} - удаление самой короткой

  stringZ=abcABC123ABCabc
  echo ${stringZ#a*C} *# 123ABCabc*
- Замена подстроки -
  ${string/substring/replacement} – *первое*
  ${string//substring/replacement} - все substring

Bash-scripts

Строки. Пример

```
OPERATION=docToPdf
SUFFIX=pdf
directory=$PWD
for file in $directory/*
do
  filename=${file%.doc}
  $OPERATION $file > "$filename.$SUFFIX"
  rm -f $file
done
```