OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

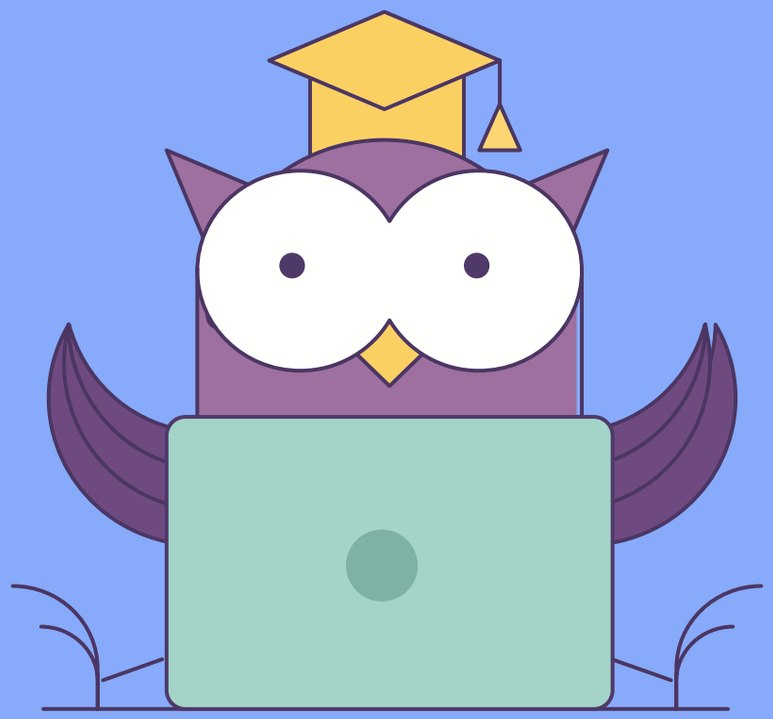
BASH

Курс «Администратор Linux»

Занятие № 4

Алексей Цыкунов Альбрехт Леонид

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо
Ставьте - если есть проблемы

Маршрут вебинара

BASH

Scripting

RegExp

SED/AWK

SH, BASH, ZSH, FISH

- **BASH** aka ‘Bourne Again Shell’ - стандартная оболочка почти во всех дистрибутивах
- **ZSH** - оболочка, практически ничем не отличающаяся от BASH. Что делает её прекрасной так это oh-my-zsh
- **FISH** - есть небольшие отличия, например, в условиях. Так же есть oh-my-fish

Запуск командной оболочки

Login shell:

- /etc/profile
- /etc/profile.d
- ~/.bash_profile
- ~/.bashrc
- /etc/bashrc

Non-login shell :

- ~/.bashrc -> /etc/bashrc -> /etc/profile.d

Помощь в BASH

- type
- help
- man
- apropos
- whatis
- info

man bash	help	info
man man	help help	
man -a intro	info info	

Man pages

Все страницы разделены на несколько категорий:

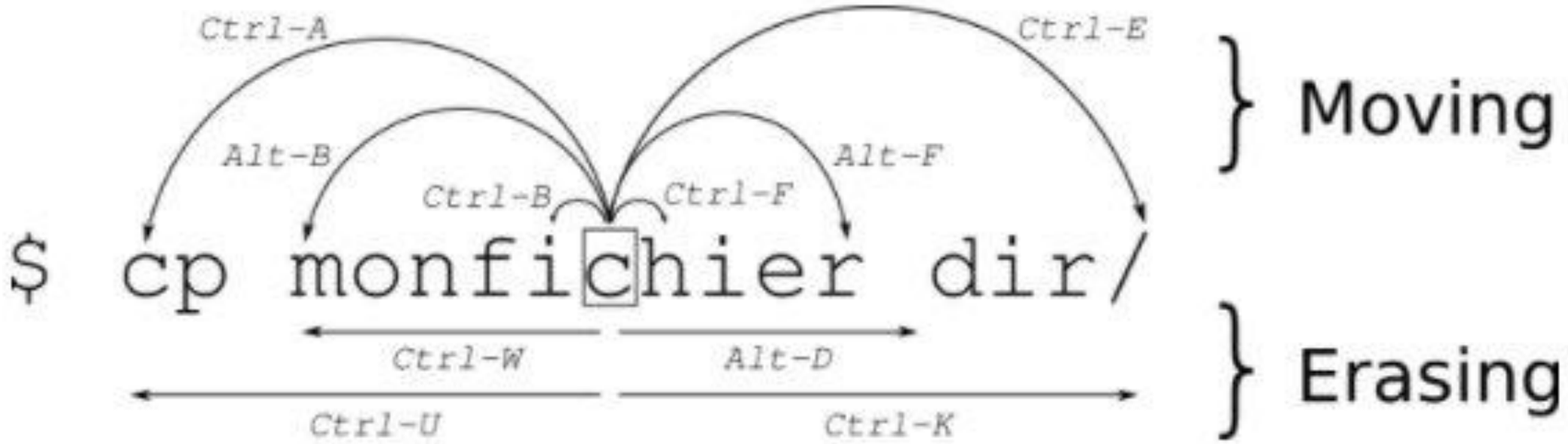
1. Основные команды
2. Системные вызовы (функции, предоставляемые ядром linux)
3. Библиотечные вызовы (функции стандартной библиотеки языка Си)
4. Специальные файлы (обычно расположенные в /dev) и драйверы
5. Форматы файлов и соглашения
6. Игры
7. Прочие страницы (также включая соглашения)
8. Команды для системного администрирования (для которых обычно требуются права суперпользователя) и демоны

[root@bash] whatis intro

Hot keys

- Tab или Ctrl+1 - автодополнение строки
- Ctrl+a Ctrl+e - возврат курсора в начало строки, переход в конец строки
- Alt+f Alt+b - перемещение на слово вперед, на слово назад
- Ctrl+j - возврат каретки (он же Enter)
- Ctrl+l - очищает экран
- Ctrl+w - удаляет слово до курсора (при этом копируя его в буфер обмена)
- Ctrl+y - вставить содержимое из буфера
- Ctrl+u - удаляет всю строку до курсора
- Ctrl+k - удаляет всю строку после курсора

Hot keys


Ctrl-A
Ctrl-E
Alt-B
Alt-F
Ctrl-B
Ctrl-F
$ cp monfichier dir/
Ctrl-W
Alt-D
Ctrl-U
Ctrl-K
Moving
Erasing

History Hot keys

- Ctrl + r - поиск по строке, повторный Ctrl + r - цикличный поиск по истории
- Ctrl + r дважды - поиск по последней поисковой строке

Во время поиска:

- Ctrl + j - закончить поиск по истории
- Ctrl + g - закончить поиск и вернуть строку к прежнему состоянию

Операторы списка

- **command1; command2** - выполняет первую, затем вторую команды
- **command1 & command2** -
- **command1 && command2** - выполняет первую, и затем выполняет вторую, только в случае если 1-ая вернула статус 0(завершилась успешно)
- **command1 || command2** - выполняет первую, и затем выполняет вторую только в случае если 1-ая вернула не 0 статус (завершилась неудачно)

Операторы списка

- **command1; command2** - выполняет первую, затем вторую команды
- **command1 & command2** - выполняет первую в фоновом режиме и одновременно выполняет вторую
- **command1 && command2** - выполняет первую, и затем выполняет вторую, только в случае если 1-ая вернула статус 0(завершилась успешно)
- **command1 || command2** - выполняет первую, и затем выполняет вторую только в случае если 1-ая вернула не 0 статус (завершилась неудачно)

Перенаправления

$ ls -l /dev/std*
lrwxrwxrwx. 1 root root 15 май 10 15:01 /dev/stderr -> /proc/self/fd/2
lrwxrwxrwx. 1 root root 15 май 10 15:01 /dev/stdin -> /proc/self/fd/0
lrwxrwxrwx. 1 root root 15 май 10 15:01 /dev/stdout -> /proc/self/fd/1

$ echo 'hello' > /dev/null 2>&1
$ echo 'hello' &> /dev/null (сокращенная форма с версии >= 4.0)
$ ls -l | grep ".log"

mkfifo myPipe
ls -l > myPipe
grep ".log" < myPipe

Перенаправления в скриптах

```
err(){
echo "E: $*" >>/dev/stderr
 }
err "My error message"

{
echo "contents of home directory"
ls ~ }
> output.txt
```

Перенаправления

<< - HERE TEXT
wc -l << EOF
Ssss
Sdsd
Sdsd
EOF

<<< - HERE STRING
$ read first second <<< "hello world"
$ echo $second $first

<<< - HERE DOC
cat << EOF > myscript.sh
#!/bin/bash
echo “Hello Linux!!!”
exit 0
EOF


BASH
THE BOURNE-AGAIN SHELL

#ШЕБАНГ! she-bang

Примеры she-bang:

- #!/usr/bin/env VAR=VALUE bash
- #!/bin/bash
- #!/bin/python
- #!/bin/perl

Варианты запуска:

- ./script.sh (необходим chmod +x)
- bash script.sh
- source script.sh

Статус возврата

- Success: команда вернула 0 статус
- Failure: команда вернула НЕ 0 статус

Статусы возврата могут быть от 0 до 255

Можно перехватить с помощью специального параметра **$?**

Переменные и параметры

- export var=value
- var=value
- var1= val
- declare var=value
- var=`ls`
- var=$(uname -r)
- var=$((2+3))
- var=$(expr 3 + 7)
- var1="${var1:-default value}"

Переменные и параметры

- export var=value
- var=value
- var1= val <- ошибка, пробел
- declare var=value
- var=`ls`
- var=$(uname -r)
- var=$((2+3))
- var=$(expr 3 + 7)
- var1="${var1:-default value}"

Переменные и параметры

- echo $var
- echo ${var}
- echo “$var”
- echo ‘$var’
- echo `$var`
- echo $($var)
- echo var

- env
- printenv
- export
- declare

Переменные и параметры

Специальные переменные:

- $@ — параметры скрипта (столбик)
- $* - все параметры скрипта (строка)
- $0 — имя скрипта
- $1 , $2 , $3 , ... — параметры скрипта, по одному
- $# — количество параметров
- $? — статус выхода последней выполненной команды
- $$ — PID оболочки
- $! — PID последней выполненной в фоновом режиме команды

Массивы

- files = $(ls) - считывается строка
- array=('first element' 'second element' 'third element')
- array=([3]='fourth element' [4]='fifth element')
- array[0]='first element'
- array[1]='second element'
- echo ${array[2]}
- IFS=$'\n'; echo "${array[*]}"

- declare -A array
- array[first]='First element'
- array[second]='Second element'

- array =(0 1 2)

Условия

if EXPR ; then команды ; fi
if EXPR ; then команды ; else другие команды ; fi
if EXPR ; then команды; elif EXPR ; then команды; else другие команды ; fi

Наглядное различие между [и [[:

```
if [ "$answer" = y -o "$answer" = yes ]
if [[ $answer =~ ^y(es)?$ ]]
```

Условия

```
case EXPR in
CASE1) команды
;& # отработать следующие команды без проверки
CASE2) команды
;;& # выполнить следующую проверку
...
CASEN) команды
;; # закончить на этом
esac
```

Условия для строк и чисел

- -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»

Проверки для файлов

- [-e FILE] — файл существует
- [-d FILE] — это директория
- [-f FILE] — это обычный файл
- [-s FILE] — размер ненулевой
- [-r FILE] — доступен для чтения
- [-w FILE] — доступен для записи
- [-x FILE] — исполняемый

Циклы

```
arr=(a b c d e f)

for i in "${arr[@]}"
do
  echo "$i"
done
```

```
arr=(a b c d e f)

for (( i=0;i<${#arr[@]};i++ ))
do
  echo "${arr[$i]}"
done
```

```
i=0
arr=(a b c d e f)

while (( $i < ${#arr[@]} ))
do
  echo "${arr[$i]}"
  ((i++))
done
```

```
i=5

until [[ i -eq 10 ]]
do
  echo "i=$i"
  i=$((i+1))
done
```

Циклы

```
#!/bin/bash
var1=1
while [ $var1 -lt 10 ]
do

  if [ $var1 -eq 5 ]
  then
    break
  fi

  echo "Iteration: $var1" var1=$(( $var1 + 1 ))
done
```

```
#!/bin/bash

for (( var1 = 1; var1 < 15; var1++ ))
do

  if [ $var1 -gt 5 ] && [ $var1 -lt 10 ]
  then
    continue
  fi

  echo "Iteration number: $var1"
done
```

It’s a trap!!!

OTUS

```
if ( set -o noclobber; echo "$$" > "$lockfile") 2> /dev/null;
then
   trap 'rm -f "$lockfile"; exit $?' INT TERM EXIT KILL
   while true
   do
       ls -ld ${lockfile}
       sleep 1
   done
  rm -f "$lockfile"
  trap - INT TERM EXIT
else
  echo "Failed to acquire lockfile: $lockfile."
  echo "Held by $(cat $lockfile)"
fi
```

Регулярные выражения

^([a-zA-Z0-9_\-\.\+]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$

BRE
- ^ - начало строки
- $ - конец строки
- . - любой символ
- \ - экранирование символа
- [A-Z] -диапазоны перечисления
- [xyz] - любой из символов
- [^xyz] - исключенные символы
- * - любое кол-во символов

Регулярные выражения

ERE - perl/egrep/awk

- {m,n} - сколько раз может встретится символ от - до
- {m} - точное кол-во встречаемости символа
- ? - символ может встретится 0 или 1 раз
- + - любое кол-во символов, но хотя бы 1
- () - группировка символов
- | - какой либо из символов

Регулярные выражения

Классы

- [:alnum:] - A-Za-z0-9.
- [:alpha:] - A-Za-z.
- [:blank:] пробел или таб
- [:cntrl:] matches control characters.
- [:digit:] - 0-9.
- [:graph:] графические символы ASCII 33 - 126
- [:lower:] a-z.
- [:print:] - [:graph:] + пробел
- [:space:] пробел и таб
- [:upper:] A-Z.
- [:xdigit:] hex 0-9A-Fa-f

Globbing and brace expansion

Bash может выполнять подстановку имен файлов
Процесс называется "globbing"
при этом используется урезанный набор регулярных выражений

- *, ?, [a-z], {a*,b*}, [^az]

```
bash$ ls -l t?.sh
bash$ ls -l [ab]*
bash$ ls -l [a-c]*
bash$ ls -l [^ab]*
bash$ ls -l {b*,c*,*est*}
bash$ echo {1..5}{0,5}%
```

awk

Скриптовый язык построчного разбора и обработки входного потока

- awk -F":" '{ print $1}' /etc/passwd
- awk -F":" '{ print NR ") username: " $1 "\tuid:" $3 }' /etc/passwd

test.awk

```
BEGIN {print "start"}
NR<10 { print FNR ") username: " $1 "\tuid:" $3 }
END {print NR}
```

- awk -f ./test.awk /etc/passwd

- FIELDWIDTHS указать ширину поля
- RS разделитель записей
- FS разделитель полей
- OFS разделитель полей вывода
- ORS разделитель записей вывода
- NF кол-во полей обрабатываемой записи
- NR номер записи
- FNR обрабатываемая запись
- IGNORECASE - игнорирование регистра

sed

sed -i ‘s/original/new/g’ file.txt:

- sed = Stream EDitor
- -i = замена (по умолчанию сохраняется оригинальный файл)
- команда:
  - s = команда замены подстроки
  - original = текст либо регулярка
  - new = текст для замены
  - g = global (или заменится только первое вхождение)
- file.txt = имя файла

Домашнее задание

Подготовить свои скрипты для решения как минимум одного из следующих кейсов:

1) watchdog с перезагрузкой процесса/сервиса
2) watchdog с отсылкой емэйла
3) анализ логов веб сервера/security лога - (на взлом/скорость ответа/выявление быстрых - медленных запросов, анализ IP адресов и кол-ва запросов от них)
4) крон скрипт с защитой от мульти запуска
5) любой скрипт на ваше усмотрение

Желательно чтобы в скрипте были:

- циклы
- условия
- регекспы
- awk
- наличие в скрипте трапов и функций

Ваши вопросы?

Заполните, пожалуйста, опрос в ЛК о занятии

Спасибо
за внимание!
До встречи в Slack и на вебинаре