Терентьев Алексей Борисович

Основы работы с командным интерпретатором bash

Материал
для лабораторных работ по курсу
ОПЕРАЦИОННЫЕ СИСТЕМЫ

Санкт-Петербургский политехнический университет Петра Великого
2016

Введение

Ввод команд

<имя_команды> <опции> <аргументы>

- **<имя_команды>** - имя исполняемого файла
- **<опции>** - дополнительные инструкции
- **<аргументы>** - объекты, с которыми работает команда

Помощь

- **help** - встроенная помощь оболочки
- **man** - система помощи Unix
- **info** - утилита для формирования текста помощи по работе с командной строкой
- **compgen -c** - вывести список доступных команд

man

man [<номер_раздела>] <имя_страницы> - находит и форматирует страницы, передавая их программе просмотра(less).

Клавиши *less*[1]:

- *PgUp/PgDown* - перемещение по тексту
- *<пробел>* - следующая страница
- *</>* - поиск строки вниз
- *<?>* - поиск строки вверх
- *<n>* - следующее вхождение искомой строки
- *<q>* - выход

[1] Горячие клавиши оболочки `http://en.wikipedia.org/wiki/Bash_(Unix_shell)`

Ввод и исполнение команд

- <\>для ввода длинных команд в несколько строк

Выполнение команд:

- <;>последовательное выполнение:
  `command1; command2`
- <**&**>выполнение в фоне:
  `command1 & command2`
- Условное выполнение:
  `command1 && command2` - выполнить 2 т. и т.т. 1 закончилась успешно
  `command1 || command2` - выполнить 2 т. и т.т. 1 закончилась неудачно

Самопроверка

Попробуйте выполнить команды:

- ```
  echo \
  foo
  ```
- `echo foo; echo 1`

Что должны вывести данные команды?:

- `true && echo yes` Ответ:
- `false && echo no` Ответ:
- `true || echo yes` Ответ:
- `false || echo yes` Ответ:
  ```

Самопроверка

Попробуйте выполнить команды:

- `echo \`
`foo`
- `echo foo; echo 1`

Что должны вывести данные команды?:

- `true && echo yes` Ответ: yes
- `false && echo no` Ответ:
- `true || echo yes` Ответ:
- `false || echo yes` Ответ: yes

Переменные, выражения, циклы

Переменные оболочки

Переменные оболочки доступны только в той оболочке, в которой были описаны

- Задание переменной:

```
VAR='value'
VAR1=0
VAR2='no'$VAR
VAR3='zero=='${VAR1}
```

- Уничтожение переменной `unset var_name`.

Переменные окружения

Переменные окружения доступны дочерним процессам

- Перевод переменной оболочки в переменную окружения:

```
VAR='value'
export VAR
```

- Как сделать глобальной?
  - **/etc/environment**
  - **/etc/profile.d/*.sh**:
  - **/.bashrc, /.profile**

```
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0
```

Вычисление выражений

- Конструкция ((...)):

```
((1+2))
VAR=$((123%34))
(( percent >= 0 && percent <= 100 ))
VAR1=$((VAR+18))
VAR2=$(($1*145)) # $1 - второй аргумент параметров командной строки
```

Как делать не нужно:

VAR = $((1+1))

обратите внимание на пробелы по бокам символа =

Ветвления

```
if <expr1>; then
   <commands1>
[elif <expr2>; then
   <commands2>]
[else
   <commands3>]
fi
```

expr:

- [-a existingfile], [-n "$nonemptstr"], [$var -lt 15]
- [[-a *.ext]], [[-n $nonempstr]],
  [[$? -eq 543 && -z $emptyestr]]
- (($var >= 123))

---

https://linuxacademy.com/blog/linux/conditions-in-bash-scripting-if-statements/

Самопроверка

Создайте переменную *bdsq* и проинициализируйте ее днем вашего рождения в квадрате. Попробуйте выполнить команду *echo $$*. Сделайте переменную доступной в дочерних процессах. Запустите bash внутри текущего. Убедитесь, что значение *echo $$* изменилось, напишите проверку, является ли полученное число четным и выведите “even” или “odd” соответственно. Ответ:

Самопроверка

Создайте переменную *bdsq* и проинициализируйте ее днем вашего рождения в квадрате. Попробуйте выполнить команду *echo $$*. Сделайте переменную доступной в дочерних процессах. Запустите bash внутри текущего. Убедитесь, что значение *echo $$* изменилось, напишите проверку, является ли полученное число четным и выведите “even” или “odd” соответственно. Ответ:

```
bdsq=$((14*14))
export bdsq
echo $$
bash echo $$
if (($bdsq%2==0)); then echo even; else echo odd; fi
```

Циклы

- for:

```
for (( i=0, j=0 ; i < 10 ; i++, j++ )); do #for i in $( ls ); do
    echo $((i*j))
done
```

- while:

```
while read a b; do
    echo $a $b
done
```

- until:

```
cnt=20
until [ $cnt -lt 10 ]; do
    echo cnt $cnt
    let cnt-=1
done
```

Самопроверка

Создайте команду, которая считывает с клавиатуры слова и выводит их до тех пор пока не будет введено слово “stop”.
Ответ:

Создайте команду, вычисляющую факториал от числа 10.
Ответ:

Самопроверка

Создайте команду, которая считывает с клавиатуры слова и выводит их до тех пор пока не будет введено слово “stop”.

Ответ:

```
 while read a; do
> echo $a
> if [[ $a == "stop"]]; then
> break
> fi
> done
```

Создайте команду, вычисляющую факториал от числа 10.

Ответ:

```
f=1; for (( i=2 ; i < 10 ; i++ )); do f=$(($f * $i)); done; echo $f
```

Подстановка команд

- ‘ ‘(апострофы):

```
cmd1 ‘cmd‘
man ‘echo man‘
echo ‘echo a \‘echo b \‘ ‘
var=‘echo a‘
```

- $(cmd):

```
cmd1 $(cmd)
man $(echo man)
echo $(echo a $(echo b))
var=$(echo a)
```

---

http://www.tldp.org/LDP/abs/html/commandsub.html

Перенаправление потоков

- Перенаправление вывода **[n]**>, **[n]**>>, **&**>, **&**>>:

```
echo str > file.f
cmd 1>> stdout.f 2>> stderr.f
cmd &> all.f
command 2>&1 >output.txt - сначала перенаправляет stderr в т
stdout,а потом stdout в output.txt(но не stderr)
```

- Перенаправление ввода <, <<:

```
read a < file.f
sort -k2 << END
> 1 apple
> 2 pear
> 3 banana
> END
```

---

http://www.ibm.com/developerworks/ru/library/l-lpic1-v3-103-4/

Конвейеризация

- Конвейеризация |:

```
cmd1 | cmd 2 p1 | cmd3 p1 | cmd4 echo 'new line' | cat file.f -
cmd 2>&1 | cmd2
```

Перенаправление только stdout в stdin

Команды bash

- Вывод содержимого файлов **cat**:

```
cat filename
```

- Вывод информации о файлах **ls**:

```
ls -la
```

- Поиск **find**:

```
find . -type f ! -perm 777
   #права доступа не 777
find . -type f -iname fname.f
   #с именем fname.f (insensitive)
find . -type f -name *.mp3 -size +10M -exec rm {} \;
   #удалить файлы *.mp3 размером больше 10 МБ
find . -type f \( -name '*.c' -or -name '*.cpp' \)
   #файлы с расширением .c или .cpp
find . -type f -printf '%s %p\n'
   #вывести список размеров и файлов
```

- **grep** поиск определенного текста в файлах:

```
grep ^a - начинаются с а
   echo Hello$'\n'world | grep ^w
grep -v [aeiou] wordlist.txt #без гласных
grep -F -x -v -f what.txt from.txt > to.txt
   #убрать строки из from, если они есть в what
```

- **awk** - поиск и замена текста:

```
awk '/123/ { print $0 }
     /abc/ { print $0 }
     /some text/ { print $0 }' samplefile
ls -l | awk '{print $5}' #5-е слово
awk '{print NR "- " $1 }' f.f #н. стр. - первое слово
awk '{print $1, $(NF-2) }' f.f #1-е и 3-е с конца
awk 'NF > 1' data.txt
   #вывести строки с более чем 1 словом
```

Самопроверка

Каким образом можно из списка файлов и папок, полученного с помощью команды ls можно извлечь только файлы функцией grep? Ответ:

Как теперь с помощью awk получить только имена файлов? Ответ:

Самопроверка

Каким образом можно из списка файлов и папок, полученного с помощью команды ls можно извлечь только файлы функцией grep? Ответ:

```
ls -l | grep ^-
```

Как теперь с помощью awk получить только имена файлов? Ответ:

```
ls -l | grep ^- | awk '{print $9}'
```

- **xargs** - запуск команды с передачей сконструированного списка аргументов(обычно список файлов от find, ls)

```
find . -mmin -30 -print0 | xargs -0 rm
   #удалить файлы, которые
      были изменены в последние полчаса
```

Команда *wc -w* подсчитывает количество слов в файле, как узнать количество слов в текущей папке? Ответ: ▯

- **xargs** - запуск команды с передачей сконструированного списка аргументов(обычно список файлов от find, ls)

```
find . -mmin -30 -print0 | xargs -0 rm
   #удалить файлы, которые
      были изменены в последние полчаса
```

Команда *wc -w* подсчитывает количество слов в файле, как узнать количество слов в текущей папке? Ответ:

```
find . -type f -print0 | xargs -0 wc -l | tail -1
```

- **head** - вывод начала файла:

```
head -85 file.f #первые 85 строк
head -c 10 f.f #первые 10 байт
```

- **tail** - вывод конца файла:

```
tail -85 file.f #последние 85 строк
tail -n +2 file.f #все, кроме первой строки
```

- **uniq** - унифицировать содержимое файлов(удалить смежные одинаковые строки, оставив только 1 экземпляр):

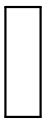
```
uniq file.f
```

- **sort** - сортировка строк в файле

```
sort file.f
sort -n file.f #численно
sort -r file.f #в обратом порядке
sort -k 3,3 file.f #по 3 слову
sort -t : -k 2,2n -k 5.3,5.4
```

Самопроверка

Как найти файл с самым большим количеством строк? Ответ:



Самопроверка

Как найти файл с самым большим количеством строк? Ответ:

```
ls -rXl | grep ^- | awk '{print $9}' | xargs wc -c | head -n -1 |
sort -nr | head -1 | awk '{print $1}'
```

Полезные ссылки

Полезные ссылки

`http://www.tldp.org/LDP/abs/html/.`

`http://ss64.com/bash/`