
Hostopia®
A DELUXE® COMPANY

Command interpreter

Bash

04-06-2014

About Bash

Bash is one of most popular varieties of Unix shell. Especially popular on Linux, where it is often used by default. Bash works in the text mode, can read commands from file, called a 'script'. it supports work with variables, control of the execution order of commands, branching and loops constructions.

Basic commands

Show help information:

man <*name*>
<*name*> **--help**

name — name of program, utility or function.

Examples:

man man
man bash
bash --help

man hier
cp --help
man mv

man rm
mkdir --help
man rmdir

Basic commands

Show system information:

uname - basic system information
lscpu - data about CPU
free - data about RAM
mount - data about mounted filesystems
df - data about usage of HDD space
ifconfig - data about network
top - data about system processes

Examples:

man free	man mount	df -h
man df	free -m	uname -a

Basic commands

Show content of file:

cat <*filename*> - get all lines
head <*filename*> - get first 10 lines
tail <*filename*> - get last 10 lines
grep <*pattern*> <*filename*> - get lines by pattern

Examples:

man cat
man head
man tail
man grep

grep login f.log
cat /etc/passwd
tail -20 text.txt
head .profile

cat /proc/cpuinfo
head -12 file.log
grep -r 'text' *
tail -f /tmp/log

Basic commands

Operations with files and dirs:

cd <*dirname*> - change dir
touch <*filename*> - create file
mkdir <*dirname*> - create dir
cp <*source*> <*destination*> - copy file or dir
mv <*source*> <*destination*> - move file or dir
rm <*filename*> - delete file
rmdir <*dirname*> - delete empty dir

Examples:

man touch	man rm	touch file.txt
man mv	rm -f file.tp*	cp file.txt /tmp/

Basic commands

Operations with permissions:

stat <*dir or file*> - display dir or file status
chmod <*mode*> <*dir or file*> - change access rules
chown <*username*> <*dir or file*> - change owner
chgrp <*grpname*> <*dir or file*> - change group

Examples:

chmod 700 file.txt
chown -R user *
chgrp users dirname
stat /tmp

chown user *.php
chgrp -R users dirname/
chmod og+rw *.txt
stat file.txt

Basic commands

Recursive search files and dirs:

find <*path*> <*parameters*>

Examples:

All .php files with perms 664 and user vic from /:
find / -name *.php -type f -perm 664 -user vic

All dirs with name .svn and group dev from path:
find /path/to/dir/ -name .svn -type d -group dev

All synlinks with name sys from current dir:
find . -name sys -type l

Basic commands

Recursive search files and dirs with handling:

find <*path*> <*parameters*> -exec <*command*> {} \;

Examples:

Set perms 664 for all .php files from /:
find / -name *.php -type f -exec chmod 664 {} \;

Check status for all dirs with name .svn from path:
find /path/ -name .svn -type d -exec stat {} \;

Remove all synlinks from current dir:
find . -type l -exec rm -f {} \;

Basic commands

Synchronization files and dirs:

rsync <*parameters*> <*source*> <*destination*>

Examples:

Sync files and dirs between 2 dirs with deletion extraneous files in /dest/:
rsync -av --delete /src/ /dest/

Sync only dir structure between 2 dirs:
rsync -av --include='*/' --exclude='*' /src/ /dest/

Basic commands

Handling text via **sed**:

sed <*parameters*> <*filename*>

Examples:

Replace all words MyISAM to InnoDB in file bkp.sql:
sed -i 's/MyISAM/InnoDB/g' bkp.sql

Show line #52 from file bkp.sql:
sed '52q;d' bkp.sql

Delete first 10 lines from file file.log:
sed -i '1,10d' file.log

Basic commands

Handling text via **awk**:

awk <*parameters*> <*filename*>

Example:

Get lines from file with 'bash' word and generate SQL Insert to database:

```
awk -F: '/bash/ {
print "Insert into Users set login=\""$1"\", uid="$3";"
}' /etc/passwd
```

Basic commands

Input/output redirection:

<command> **>** *<filename>* - writing to file
<command> **<** *<filename>* - reading from file
<command> **|** *<command>* - conveyor belt of commands, each previous command passes result to next command for handling.

Examples:

ls -l /home/ > /tmp/list.txt
mysql -u user -p database < tables.sql
cat /etc/passwd | grep '/bin/bash' | grep user > u.txt

First steps

Requirements for bash scripts:

- All bash scripts shoud begin with 2 following lines:

#!/bin/bash
<empty line>

Example (showing the text)**:**

#!/bin/bash

echo 'This is a text'

First steps

- Decency in bash scripting - to use extension .sh for bash scripts (this means '*ba**sh***' or '***sh**ell*'):

Example:

Example.sh

First steps

- Bash script should be with permissions for executing:

Example:

rwxr--r-- (744) Example.sh

Executing permission for owner only

Basic operations

Work with variables:

Assigning: *<variable>=<value>*
Using: *$<variable>*

Examples (operations with strings)**:**

```
A='This is a text'
echo $A
```

```
CMD=`echo 'This is a text'`
echo $CMD
```

```
A='is'
B="This $A a text"
echo $B
```

```
A='is'
CMD=`echo "This $A text"`
echo $CMD
```

Basic operations

Work with variables:

Examples (arithmetic operations)**:**

```
A=3
echo $(( $A + 2 ))

A=6
B=$(( $A + 4 ))
echo $(( $B / 2 ))

echo $(( 4 / 2 + 6 - 3))
```

Basic constructions

Construction **if**:

if <*condition(s)*>
then
    <*command(s)*>
fi

if <*condition(s)*>
then
    <*command(s)*>
else
    <*command(s)*>
fi

if <*condition(s)*>
then
    <*command(s)*>
elif <*condition(s)*>
then
    <*command(s)*>
elif <*condition(s)*>
then
...
else
    <*command(s)*>
fi

Basic constructions

Condition for ***files***:
[!] -a file — true if file [don't] exists
[!] -d file — true if file [don't] exists and it's a dir
[!] -f file — true if file [don't] exists and it's a file
[!] -h file — true if file [don't] exists and it's a symlink
[!] -r file — true if file [don't] exists and it's readable
[!] -w file — true if file [don't] exists and it's writable
[!] -x file — true if file [don't] exists and it's executable

Condition for ***strings***:
-z string — true if length of string is 0
-n string — true if length of string is not 0
string1 = string2 — true if strings is equal
string1 != string2 — true if strings is not equal

Basic constructions

If examples:

```
A=3
if [ "$A" = '3' ]
then
    echo 'A = 3'
fi

if [ ! -f '/etc/sudoers' ]
then
    echo "File don't exists"
fi
```

```
if [ -d '/tmp' ]
then
    echo "Dir exists"
else
    echo "Dir don't exists"
fi
```

Basic constructions

Logical AND, logical OR:

<condition> **&&** <condition>
<condition> **||** <condition>

Example:

```
X=3
Y=2
if ([ "$X" = '5' ] && [ ! "$Y" = '2' ]) || [ "$X" = '3' ]
then
    echo 'One of conditions is true'
fi
```

Basic constructions

Construction **case**:

```
case <value> in
<pattern> )
    <command(s)>
;;
<pattern> )
    <command(s)>
;;
...
<pattern> )
    <command(s)>
;;
esac
```

Example:

```
A=2
case $A in
1 )
        echo "= 1"
;;
2 )
        echo "= 2"
;;
* )
        A=4
;;
esac
```

Basic constructions

Construction **for**:

for <*variable*> in <*values*>
do
    <*command(s)*>
done

Examples:

```
for x in {0..5}
do
    echo $x
done
```

```
for x in $(ls /tmp/)
do
    stat /tmp/$x
done
```

Basic constructions

Construction **while**:

while <*condition(s)*>
do
    <*command(s)*>
done

Example:

```
N=0
while [ "$N" != "10" ]
do
    echo "N = $N";N=$((N + 1))
done
```

Basic constructions

Construction **until**:

until <*condition(s)*>
do
    <*command(s)*>
done

Example:

```
N=0
until [ "$N" = "10" ]
do
    echo "N = $N";N=$((N + 1))
done
```

Basic constructions

Input parameters:

<scriptname> <param1> <param2> … <paramN>

$0 — script name
$1 … $N — parameters
$# - count of parameters (except $0)

Example:

Example.sh /tmp/file.xml
$0 = Example.sh
$# = 1
$1 = /tmp/file.xml

Basic constructions

User functions:

Declaration:

function <*name*>()
{
    <*command(s)*>
}

Using:

<*functionname*>

Example:

```
function say_hello()
{
    local A='Hello'
    echo $A
}

say_hello
```

Thank you

Eugene Zaporozhets