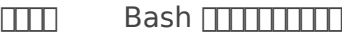




- if ☐ ☐ _____
- test ☐ _____
- ☐ ☐ ☐ _____
- ☐ ☐ ☐ _____
- ☐ ☐ ☐ ☐ _____
- ☐ ☐ ☐ _____
- ☐ ☐ ☐ _____
- test ☐ ☐ ☐ ☐ ☐ ☐ _____
- ☐ ☐ ☐ _____
- ☐ ☐ ☐ ☐ ☐ ☐ ☐ _____
- case ☐ ☐ _____
- ☐ ☐ ☐ _____



if

```
if [ ]
```

```
if commands; then
    commands
[elif commands; then
    commands...]
[else
    commands]
fi
```



if [] then
[]

elif []

else

```
if test $USER = "foo"; then
    echo "Hello foo."
else
    echo "You are not foo."
fi
```

[]

\$USER []

foo []

Hello foo. []

if [] then []
[]

Bash

```
if true
then
    echo 'hello world'
fi

if false
then
    echo 'it is false' # [ ]
fi
```

[] true [] false []
[] if false []

if true

[] if []

```
$ if true; then echo 'hello world'; fi
hello world

$ if false; then echo "It's true."; fi
```

[] if []

0 []

```
$ if echo 'hi'; then echo 'hello world'; fi
hi
hello world
```

[] if [] echo 'hi' []

0 []

then []

expression test 0 test

1 []

```
$ test -f /etc/hosts
```

```
$ echo $?
```

0

```
$ [ -f /etc/hosts ]
```

\$ echo \$?

0

test

/etc/hosts

0

The diagram shows a memory stack with a pointer and a test label. The pointer is represented by a box containing a bracket and a small box, pointing to a specific memory location. The test label is represented by a box containing the text "test". Below the pointer and test label is a long horizontal bar representing memory, divided into 20 segments. The pointer points to the 10th segment from the left.

[illegible]#

```
if test -e /tmp/foo.txt ; then
```

```
echo "Found foo.txt"
```

fi


```
if [ -e /tmp/foo.txt ] ; then
```

```
echo "Found foo.txt"
```

fi


```
if [[ -e /tmp/foo.txt ]] ; then
```

```
echo "Found foo.txt"
```

fi

--	--	--	--	--

`if` `test`

XXXXXXXXXX

XXXX

XXXXXXXXXX

- [-b file] file XXXXXXXXXX true
 - [-c file] file XXXXXXXXXXXX true
 - [-d file] file XXXXXXXX true
 - [-e file] file XXXX true
 - [-f file] file XXXXXXXXXX true
 - [-g file] file XXXXXX ID true
 - [-G file] file XXXXXXXX ID true
 - [-h file] file XXXXXXXXXX true
 - [-k file] file XXXXXXXX "sticky bit" true
 - [-L file] file XXXXXXXXXX true
 - [-N file] file XXXXXXXXXXXX true
 - [-O file] file XXXXXXXXXX ID true
 - [-p file] file XXXXXXXXXX true
 - [-r file] file XXXXXXXXXXXX true
 - [-s file] file XXXXXXXXXX true
 - [-S file] file XXXXXXXX socket true
 - [-t fd] fd XXXXXXXXXXXX true
- XXXXXXXXXXXXXXXXXXXX
- [-u file] file XXXXXX setuid true
 - [-w file] file XXXXXXXXXXXX true
 - [-x file] file XXXXXXXXXXXX true
 - [FILE1 -nt FILE2] FILE1 FILE2 XXXXXXXX FILE1 FILE2
XXXX true
 - [FILE1 -ot FILE2] FILE1 FILE2 XXXXXXXX FILE2 FILE1
XXXX true
 - [FILE1 -ef FILE2] FILE1 FILE2 XXXXXXXX inode true

XXXXXX

```
#!/bin/bash

FILE=~/.bashrc

if [ -e "$FILE" ]; then
    if [ -f "$FILE" ]; then
        echo "$FILE is a regular file."
    fi
fi
```

```

if [ -d "$FILE" ]; then
    echo "$FILE is a directory."
fi
if [ -r "$FILE" ]; then
    echo "$FILE is readable."
fi
if [ -w "$FILE" ]; then
    echo "$FILE is writable."
fi
if [ -x "$FILE" ]; then
    echo "$FILE is executable/searchable."
fi
else
    echo "$FILE does not exist"
    exit 1
fi

```

```

##### $FILE ##### $FILE ##### $FILE
##### [ -e $FILE ] [ -e ] $FILE [ -e "$FILE" ]
[ [ -e "" ]

```

```

#####

```

```

#####

```

- [string] string 0
- [-n string] string
- [-z string] string
- [string1 = string2] string1 string2
- [string1 == string2] [string1 = string2]
- [string1 != string2] string1 string2
- [string1 '>' string2] string1 string2
- [string1 '<' string2] string1 string2

```

[ test > <
#####

```

shell

```

#####

```

```

#!/bin/bash

```

```

ANSWER=maybe

```

```
$ANSWER There is no answer.
```

```
yes
```

```
[ -n "$COUNT" ] test
```

```
[ -n " " ]
```

--	--	--	--

- | | | | |
|-----------------------------|----------|----------|------|
| • [integer1 -eq integer2] | integer1 | integer2 | true |
| • [integer1 -ne integer2] | integer1 | integer2 | true |
| • [integer1 -le integer2] | integer1 | integer2 | true |
| • [integer1 -lt integer2] | integer1 | integer2 | true |
| • [integer1 -ge integer2] | integer1 | integer2 | true |
| • [integer1 -gt integer2] | integer1 | integer2 | true |

```
INT=-5

if [ -z "$INT" ]; then
```

```

echo "INT is empty." >&2
exit 1
fi
if [ $INT -eq 0 ]; then
    echo "INT is zero."
else
    if [ $INT -lt 0 ]; then
        echo "INT is negative."
    else
        echo "INT is positive."
    fi
    if [ $((INT % 2)) -eq 0 ]; then
        echo "INT is even."
    else
        echo "INT is odd."
    fi
fi

```

■■■■■■■■■■

\$INT ■■■■■■■■■■

0 ■■■■■■■■■■■■■■■■■■■■■■

■■■■

[[expression]] ■■■■■■■■■■■■

[[string1 =~ regex]]

■■■■■

regex ■■■■■■

=~ ■■■■■■

■■■■■

```
#!/bin/bash
```

```
INT=-5
```

```

if [[ "$INT" =~ ^-?[0-9]+$ ]]; then
    echo "INT is an integer."
    exit 0
else
    echo "INT is not an integer." >&2
    exit 1

```

test

--	--	--	--	--	--	--

- AND ☐☐☐☐ && ☐☐☐☐☐☐☐☐
- OR ☐☐☐☐ || ☐☐☐☐☐☐☐☐ -a ☐
- NOT ☐☐☐☐☐☐ ! ☐ -o ☐

AND

```
#!/bin/bash
```

MIN_VAL=1

MAX_VAL=100

INT=50

```
if [[ "$INT" =~ ^-?[0-9]+$ ]]; then
    if [[ $INT -ge $MIN_VAL && $INT -le $MAX_VAL ]]; then
        echo "$INT is within $MIN_VAL to $MAX_VAL."
    else
        echo "$INT is out of range."
    fi
else
    echo "INT is not an integer." >&2
    exit 1
fi
```

\$MAX_VAL

!

```
if [ ! \( $INT -ge $MIN_VAL -a $INT -le $MAX_VAL \) ]; then
    echo "$INT is outside $MIN_VAL to $MAX_VAL."
else
```

```
    echo "$INT is in range."
fi
```

Bash

□ □ -a □ □ □ □ □ □ □ □ □ □ □ □ □ □ && □ □ □ □ □ □ □ □ □ □ □ □ □ □

```
if !([ $INT -ge $MIN_VAL ] && [ $INT -le $MAX_VAL ]); then
    echo "$INT is outside $MIN_VAL to $MAX_VAL."
else
    echo "$INT is in range."
fi
```

--	--	--	--

Bash [] [] [] ((...)) [] [] [] [] [] [] [] [] [] []

```
if ((3 > 2)); then
    echo "true"
fi
```

[illegible]

[illegible]

```
$ if ((1)); then echo "It is true."; fi
It is true.
$ if ((0)); then echo "It is true."; else echo "it is false."; fi
It is false.
```

((1))

 ((0))

```
$ if (( foo = 5 ));then echo "foo is $foo"; fi
foo is 5
```

[illegible]

XXXXXXXXXXXXXXXXXXXX

0 XXXXXXX

```
$ if (( foo = 0 ));then echo "It is true.";else echo "It is false."; fi
It is false.
```

XXXXXXXXXXXXXXXXXXXX

```
#!/bin/bash
```

```
INT=-5
```

```
if [[ "$INT" =~ ^-[0-9]+$ ]]; then
    if ((INT == 0)); then
        echo "INT is zero."
    else
        if ((INT < 0)); then
            echo "INT is negative."
        else
            echo "INT is positive."
        fi
        if (( (INT % 2) == 0 )); then
            echo "INT is even."
        else
            echo "INT is odd."
        fi
    fi
else
    echo "INT is not an integer." >&2
    exit 1
fi
```

XXXXXXXXXXXX

((...)) XXXXX

Bash XXXXXXX

XXXXXXXXXXXX

if XXXXX

test XXXXXXXXXXXXXXX

((...)) XXXXX

test

XXXXXXXXXXXXXXXXXXXX

Bash XXXXXXX

&& AND || OR

XXXXXXXXXXXXXXXXXXXX

```
$ command1 && command2
```

```
$ command1 || command2
```

|| && command1 command1 command2 ||
command1 command1 command2

```
$ mkdir temp && cd temp
```

temp

```
$ [ -d temp ] || mkdir temp
```

temp

```
[ ! -d temp ] && exit 1
```

temp 1

if &&

```
if [ condition ] && [ condition ]; then  
    command  
fi
```

```
#!/bin/bash
```

```
filename=$1
```

```
word1=$2
```

```
word2=$3
```

```
if grep $word1 $filename && grep $word2 $filename  
then  
    echo "$word1 and $word2 are both in $filename."  
fi
```

word1 word2 if

&& if

```
[[ -d "$dir_name" ]] && cd "$dir_name" && rm *

# 测试

if [[ ! -d "$dir_name" ]]; then
    echo "No such directory: '$dir_name'" >&2
    exit 1
fi
if ! cd "$dir_name"; then
    echo "Cannot cd to '$dir_name'" >&2
    exit 1
fi
if ! rm *; then
    echo "File deletion failed. Check results" >&2
    exit 1
fi
```

case 语句

```
case 表达式 in
    模式1)
        命令1 ;;
    模式2)
        命令2 ;;
    ...
esac
```

elif 或 if

```
case expression in
    pattern )
        commands ;;
    pattern )
        commands ;;
    ...
esac
```

表达式 模式 命令

```
#!/bin/bash
```

```
echo -n "请输入3个字符: "
read character
case $character in
    1 ) echo 1
```

```

;;
2 ) echo 2

;;
3 ) echo 3

;;
* ) echo 
esac

```

```


```

```

*

```

```

if

```

```

else

```

```


```

```

#!/bin/bash

```

```

OS=$(uname -s)

```

```

case "$OS" in

```

```

    FreeBSD) echo "This is FreeBSD" ;;

```

```

    Darwin) echo "This is Mac OSX" ;;

```

```

    AIX) echo "This is AIX" ;;

```

```

    Minix) echo "This is Minix" ;;

```

```

    Linux) echo "This is Linux" ;;

```

```

    *) echo "Failed to identify this OS" ;;

```

```

esac

```

```


```

```

case

```

- a) a

- a|b) a b

- [[:alpha:]]

- ???) 3

- *.txt) .txt

- *)

```

case

```

```

#!/bin/bash

```

```

echo -n "> "

```

```

read character

```

```

case $character in

```

```

[[[:lower:]] | [[[:upper:]] ] echo "    $character"

;;

[0-9] ) echo "    $character"

;;

* ) echo "    "

esac

```

Bash 4.0

case

case

Bash 4.0

;;&

```
#!/bin/bash
```

```
# test.sh
```

```
read -n 1 -p "Type a character > "
```

```
echo
```

```
case $REPLY in
```

```
    [[[:upper:]]]) echo "'$REPLY' is upper case." ;;&
```

```
    [[[:lower:]]]) echo "'$REPLY' is lower case." ;;&
```

```
    [[[:alpha:]]]) echo "'$REPLY' is alphabetic." ;;&
```

```
    [[[:digit:]]]) echo "'$REPLY' is a digit." ;;&
```

```
    [[[:graph:]]]) echo "'$REPLY' is a visible character." ;;&
```

```
    [[[:punct:]]]) echo "'$REPLY' is a punctuation symbol." ;;&
```

```
    [[[:space:]]]) echo "'$REPLY' is a whitespace character." ;;&
```

```
    [[[:xdigit:]]]) echo "'$REPLY' is a hexadecimal digit." ;;&
```

```
esac
```

```
$ test.sh
```

```
Type a character > a
```

```
'a' is lower case.
```

```
'a' is alphabetic.
```

```
'a' is a visible character.
```

```
'a' is a hexadecimal digit.
```

;;&

case



- [The Linux Command Line](#), William Shotts
-

☐☐ #2

☐ cocobook-sync ☐☐ 2026-10-28 10:52:09 UTC

☐ cocobook-sync ☐☐ 2026-10-28 11:27:28 UTC