

Shell Scripting Cheatsheet

BASH

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

Constructs

if-elif-else

```
if [[ "$INPUT" == "date" || "$INPUT" == "DATE" ]]; then
    date
elif [ "$INPUT" == "" ]; then
    echo "Given a blank input."
else
    echo "I do not know what to do."
fi
```

switch-case

```
case "$VAR" in
    1)
        echo 1
        ;;
    2|two)
        echo -e "Two\n"
        ;;
    *) # else/catch-all
        echo 'No option available!'
        exit 1
        ;;
esac
```

switch-case (with getopt)

```
GOPTS=$(getopt -o h1 --long help -- "$@")
eval set -- "$GOPTS"
while true; do
    case "$1" in
        1)
            echo 1
            shift
            ;;
        -h|--help)
            echo -e "Help\n"
            shift
            ;;
        *) # getopt error
            echo 'GETOPTS ERROR!'
            exit 1
            ;;
    esac
done
```

Loops

for

```
for i in $( ls ); do echo $i; done
```

while

```
while [ 1 == 1 ]; do
    date
    break
done
```

until

```
COUNTER=0
until [ "$COUNTER" == 10 ]; do
    echo "$COUNTER"
    let COUNTER=COUNTER+1
done
```

Function

```
function copyfstab {
    cat /etc/fstab > ~/fstab.txt
}
```

POSIX Shell

```
#!/bin/sh
```

Constructs

if-elif-else

```
if test "$INPUT" == "date" ||
"$INPUT" == "DATE"; then
    date
elif [ -z "$INPUT" ]; then
    echo 'Given a blank input.'
else
    echo 'I do not know what to do.'
```

if-else (one-liner)

```
[ -f /etc/hosts ] && echo "Found" ||
echo "Not found"
```

switch-case

```
case "$VAR" in
    1)
        echo 1
        ;;
    2|two)
        printf "Two\n"
        ;;
    *) # else/catch-all
        echo 'No option available!'
        exit 1
        ;;
esac
```

Loops

for

```
LIST=$(ls)
for i in $LIST; do
    echo "$i"; printf '\n'
done
```

while

```
while test 1 == 1; do
    date
    break
done
```

until

```
COUNTER=0
until test "$COUNTER" == 10; do
    echo "$COUNTER"
    COUNTER=$((COUNTER + 1))
done
```

Function

```
function copyfstab {
    cat /etc/fstab > ~/fstab.txt
}
```

Integer Comparison

- eq - is equal to
- ne - is not equal to
- gt - is greater than
- ge - is greater than or equal to
- lt - is less than
- le - is less than or equal to
- < - is less than
- <= - is less than or equal to
- > - is greater than
- >= - is greater than or equal to

```
if (( 1 < 2 )); then echo "True"; fi
if (( 1 -lt 2 )); then echo True; fi
```

String Comparison

- = - is equal to
- == - is equal to
- != - is not equal to
- > - is greater than in ASCII alphabetical order
- < - is less than in ASCII alphabetical order
- z - zero length (null)
- n - string is not null

NOTE: "=" is the same as "==" when the statement is in single brackets

NOTE: Double-brackets enable extra features such as wildcards

```
[[ $y == x* ]] - wildcard matching
[[ $y == "x*" ]] - Literal x*
```

Conditions

- b - file is a block device
- c - file is a character device like a terminal (tty) device file.
- d - file is a directory
- e - file exists
- f - file is a regular file (not a directory or device file)
- G - group-id of file same as user
- g - set-group-id (sgid) flag set on file or directory
- h - file is a link
- k - sticky bit set
- L - file is a symbolic link
- N - file modified since it was last read
- O - user owns file
- p - file is a pipe
- r - read permission
- S - file is a socket
- s - file is not zero size
- u - set-user-id (suid) flag set on file
- w - file has write permission
- x - file has execute permission

NOTE: "!" can be used as "NOT" to negate the condition

```
if [[ ! -e ./Linux.odt ]]; then
    echo True
fi
```

Conditions Supported by "test"

NOTE: "test" also supports the condition flags list under "Conditions", "Integer Comparison", and "String Comparison".

- nt - file1 is newer than file2
- ot - file1 is older than file2
- ef - file1 is another name for file2 (i.e. a link)

Integer Comparison

- \$@ - Script's parameters
- * - Script's parameters
- \$# - Number of parameters
- \$? - Last command/program exit-status
- \$ - Current option flag
- \$\$ - Process ID of shell
- \$! - Process ID of recent background process
- \$0 - Shell or script name

Tips and Tricks

Get Mesa Version (one-liner)

```
sudo dpkg -s libgles2-mesa | grep
'^Version'
```

See

http://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html