Exit Status

What You Will Learn

- How to check the exit status of a command.
- How to make decisions based on the status.
- How to use exit statuses in your own scripts.

Exit Status / Return Code

- Every command returns an exit status
- Range from 0 to 255
- 0 = success
- Other than 0 = error condition
- Use for error checking
- Use man or info to find meaning of exit status

Checking the Exit Status

- $? contains the return code of the previously executed command.

```
ls /not/here
echo "$?"
```

Output:

2

```
HOST="google.com"
ping -c 1 $HOST
if [ "$?" -eq "0" ]
then
  echo "$HOST reachable."
else
  echo "$HOST unreachable."
fi
```

```
HOST="google.com"

ping -c 1 $HOST

if [ "$?" -ne "0" ]
then
  echo "$HOST unreachable."
fi
```

```
HOST="google.com"
ping -c 1 $HOST
RETURN_CODE=$?

if [ "$RETURN_CODE" -ne "0" ]
then
  echo "$HOST unreachable."
fi
```

&& and ||

- && = AND

```
mkdir /tmp/bak && cp test.txt /tmp/bak/
```

- || = OR

```
cp test.txt /tmp/bak/ || cp test.txt /tmp
```

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST && echo "$HOST reachable."
```

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST || echo "$HOST unreachable."
```

The semicolon

- Separate commands with a semicolon to ensure they all get executed.

```
cp test.txt /tmp/bak/ ; cp test.txt /tmp
# Same as:
cp test.txt /tmp/bak/
cp test.txt /tmp
```

Exit Command

- Explicitly define the return code
  - exit 0
  - exit 1
  - exit 2
  - exit 255
  - etc...
- The default value is that of the last command executed.

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST
if [ "$?" -ne "0" ]
then
   echo "$HOST unreachable."
   exit 1
fi
exit 0
```

Summary

- All command return an exit status
- 0 - 255
- 0 = success
- Other than 0 = error condition
- $? contains the exit status
- Decision making - if, &&,||
- exit

Exit Status

DEMO