Bash Scripting Part 3

OPS102 Week 9 Class 1

Chris Tyler/John Sellens

July 11, 2024

Seneca Polytechnic

Outline

Recap From Last Class

Recap From Last Class

- Reading input
- Command capture / command substitution
- Integer arithmetic
- Exit status
- if statements

The Test Command

The test Command

- The `test` command (which is a bash builtin) can perform a variety of comparisons and tests.
- It returns success (0 exit status) if the test succeeds, or non-zero otherwise.
- For example:

```
test "$NAME" == "Chris"
```

- `test` is very often used with if/then/fi:

```
if test "$name" == "Chris"
then
  superpowers="Yes"
fi
```

The test command and [[ ]]

- The double square bracket is an aliase for the `test` command. When you use this alias, an extra argument with the closing square brackets is required:

```
if [[ "$name" == "Chris" ]]
then
  superpowers="Yes"
fi
```

- Single square bracket is supported in all Posix shells; double square bracket provides an improved version of the `test` command in Bash.

/usr/bin/test and [ ]

- Besides the bash builtin `test` (or `[[ ]]`), there are two other Posix-compatible versions of `test` available:
  - External version of `test`: `/usr/bin/test` or `/usr/bin/[`
  - Shell builtin: `[`
- These versions of `test` have a slightly different syntax. For simplicity, we won't be using them in this course. Refer to the corresponding man pages (test(1) or bash(1)) for additional information if you're interested.

Tests 1: Filesystem entries (Files/Dirs/Links)

- This first group of tests deals with filesystem entries, such as files and directories. Each test expects one argument, a pathname:
  - `-f pathname` – is pathname a regular file?
  - `-d pathname` – is pathname a directory?
  - `-l pathname` – is pathname a symbolic link?
- These tests check if the pathname exists and is of a certain type. Also:
  - `-e pathname` – does pathname exist?
  - `-s pathname` – does pathname exist and have a size greater than zero?

Tests 2: File Permissions

- The next group of tests deals with pathname permissions.
- Each test expects one argument, a pathname:
  - `-r pathname` – is pathname readable?
  - `-w pathname` – is pathname writable?
  - `-x pathname` – is pathname executable?
- These tests check if the pathname exists and has that permision for the current user.

Tests 3: Strings

- These tests accept two string arguments, which are compared:
  - `string1 == string2` – do strings match?
  - `string1 != string2` – do strings not match?
  - `string1 > string2` – is string1 greater than (sorts after) string2?
  - `string1 < string2` – is string1 less than (sorts before) string2?
- With `test`, you will usually want to quote the strings and < and >.
- For the == and != operators, string2 can be a glob pattern.
- Also:
  - `-z string1` – is string1 length equal to 0?
  - `-n string1` – is string1 length greater than 0?

Tests 4: Numeric Comparisons

- These tests accept two integer arguments, which are compared:
  - `integer1 -eq integer2` – are the integers equal?
  - `integer1 -ne integer2` – are the integers not equal?
  - `integer1 -gt integer2` – is integer1 greater than integer2?
  - `integer1 -ge integer2` – is integer1 greater than or equal to integer 2?
  - `integer1 -lt integer2` – is integer1 less than integer2?
  - `integer1 -le integer2` – is integer1 less than or equal to integer2?
- These are the numeric, rather than string, comparison operators.

All the Rest of the Tests

- These are just the most commonly-used tests.
- See the bash(1) manpage for other tests that might be useful.

Negating and Combining Tests

- You can negate (invert) a test with the ! operator:

```
[[ ! -f "$F" ]]  # check that $F isn't a regular file
```

- You can combine tests using the && (logical and) and || (logical or) operators:

```
[[ $A -eq $B && $C -eq $D ]]
[[ $X -eq $Y || $X -eq $Z ]]
```

- This should, of course, look familiar! Just like C!

Using Tests

- Remember to quote arguments which include whitespace separators.
- Be careful with the < and > comparison operators – if you have a syntax error, you may accidentally redirect input or output.
  - Which in the case of > may truncate and destroy a file!

Examples of using test: Strings

```
#!/bin/bash
architecture="$(uname -m)" # uname gets system information

if [[ "$architecture" == "x86_64" ]]
then
  echo "Your computer architecture is Intel/AMD x86_64."
elif [[ "$architecture" == "aarch64" ]]
then
  echo "Your computer uses the 64-bit Arm architecture."
else
  echo "Your computer uses an unrecognized architecture."
fi

exit 0
```

Examples of using test: Integer Numbers

```
#!/bin/bash
read -p "Enter the customer's date of birth: " birth
# Calculate the time in seconds that the customer turns/tuned 19
ageseconds="$(date -d "$birth + 19 years" +%s)"
# See if the current time in seconds is greater than that
now="$(date +%s)"

# Tell the user if the customer is old enough to be served alcohol
if [[ "$ageseconds" -lt "$now" ]]; then
  echo "The customer is of legal drinking age in Ontario."
else
  echo "The customer is too young to legally drink in Ontario."
fi
exit 0
```

Examples of using test: Integer Numbers

```
#!/bin/bash

coinflip=$((RANDOM % 2))

if [[ "$coinflip" == 0 ]]; then
  echo "Heads!"
else
  echo "Tails!"
fi

exit "$coinflip"
```

Examples of using test: File and Permissions

```
#!/bin/bash
read -p "Enter the file to be deleted: " f
if [ ! -f "$f" ]; then
  echo "The filename '$f' does not refer to a regular file - skipping."
elif [ ! -r "$f" ]; then
  echo "The file '$f' is not readable (by you) - skipping."
else
  read -p "Delete the regular file '$f'? (Y/N): " YESNO
  if [[ "$YESNO" == "Y" || "$YESNO" == "y" || "$YESNO" == "Yes"
    || "$YESNO" == "yes" || "$YESNO" == "YES" ]]; then
    echo "Deleting the file '$f'..."
    rm "$f"
    echo "...done."
  else
    echo "Skipping the file '$f' as requested."
  fi
fi
exit 0
```

Summary

Summary

- The `test` command has many variants.
- And is key to decision making in scripts.
- Exit status is the key.