# Bash Scripting Workshop 1

How to get the most out of bash

Todd Moore

July 25, 2019

## Unix

## How did programs input data originally?

Originally, before UNIX, programs had to explicitly connect to input and output devices. Specific details of how a punchcard reader, for example, would be required to load input for a particular program. This was obviously a pretty tedious task.

# Punchcard

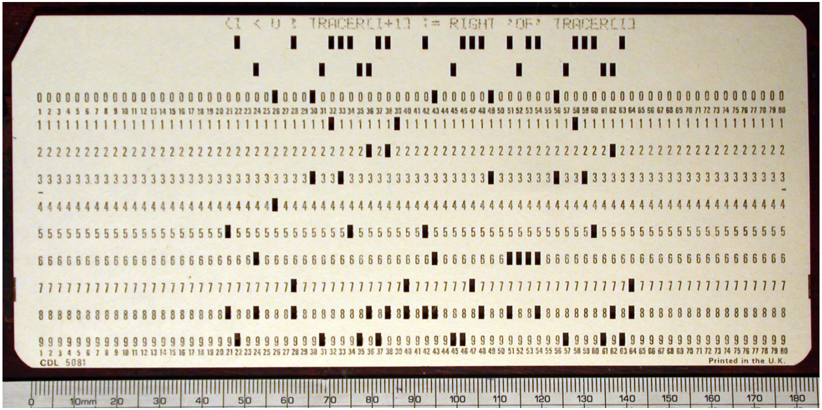

Figure: punchcard

## Unix Abstract devices

With the advent of abstract devices in UNIX, the above problem was solved and data input was abstracted into a stream of bytes with an EOF (End of File) character.

## Standard Input

- Standard input is stream data (often text) going into a program
- It's file descriptor is **0** (zero)

## Standard Output

- Standard output is the stream where a program writes its output data.
- Not all programs generate output.
- One example of standard out is the list of files from ls
- Or if you ˜echo “hello, world”, “hello, world” is written to standard out
- The file descriptor is **1**.

## Standard Error

- Standard error is another output stream typically used by programs to output error messages or diagnostics.
- It's mostly used to differentiate between normal output and output that represents an error.
- It's pretty much the same as standard output except for that it's written to a different stream for differentiation.
- The file descriptor is **2**

## Pipelines

What is a pipeline?

- A pipeline is a sequence of one or more commands separated by one of the control operators | or |&.
- The output of each command in the pipeline is connected via a pipe to the input of the next command. That is, each command reads the previous command's output.

```
$ git log | grep "v1.0.0"
```

## Redirection

What does `2>&1` mean?
You may oten see the above in scripts usually in the format of something like > `/dev/null 2>&1`
This might seem a bit cryptic at first, but it essentially means is send all output of the running program to the blackhole `/dev/null` and include both `standard error` & `standard output`. You may want to only send standard output to dev null and that would be `1>/dev/null`. Or maybe just standard error `2>/dev/null`

# Redirection Illustrated

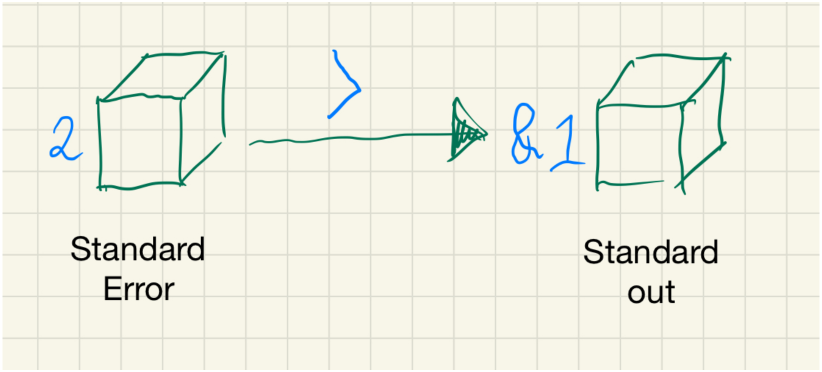

Figure: redirection

## Exercise 1.1

```
#!/usr/bin/env bash

set -e

redirect_to_std_out() {
  # 1. Change this to redirect to stderr
  echo "This is an error" >&1 | sed  "s/ is/ isn't/"

}

main() {
  redirect_to_std_out
}

main
```

## Bash

## Shebang! hashbang! pound-bang!

- The shebang is an interpreter directive
- It instructs the shell which interpreter to use to parse the script.

Examples:

```
#!/bin/sh
#!/usr/bin/env python
#!/bin/false
```

## The Set Builtin

Have you ever wondered what the following means

```
#!/usr/local/env bash

set -e # What does this mean
```

`set` allows you to change the values of shell options and set the positional parameters, or to display the names and values of shell variables

If no options or arugments are supplied, `set` displays the names and values of all shell variables and functions.

## Exercise 2.0

```
#!/usr/bin/env bash

# We want to see all the parameters
# and options that are set in our shell,
# how do we display these?

2_0() {
  echo "Hello, world"
}
```

## Answer 2.0

```
#!/usr/bin/env bash

# We want to see all the parameters
# and options that are set in our shell,
# how do we display these?
+ set

2_0() {
  echo "Hello, world"
}
```