Fundamentals of scripting

A script is just a “program” made up of shell commands. A shell script is a text file with the following characteristics:

- its first line is 
  
  ```bash
  #!/bin/bash
  ```
  indicating the shell to use to execute the script,

- it has execute permission.

(It is also possible to execute the script by invoking `bash` on the script, as in `bash scriptfile`.) A script is executed in a subshell.

To exit a script with a specific exit code \( n \), use the command `exit n`.

When a script is invoked, the shells sets a number of special variables to hold the arguments passed to the script. The name of the script is held in variable \( \$0 \). The arguments to the script are held in variables \( \$1 \), \( \$2 \), \( \$3 \), and so on. For argument number 10 and above, use braces (\( \$\{10\} \)). These variables are called positional parameters. The special variable \( \$* \) holds the list of arguments. The special variable \( \$\# \) holds the number of arguments passed to the shell. (If no arguments are passed, \( \$\# \) is null.)

Positional parameters cannot be set the way normal variables can. To reset the value of positional parameters, you use the `set` command: the command `set word1 word2 word3` resets the positional parameters as though `word1`, `word2` and `word3` had been passed as arguments to the script. Here is an example. Assume you have a script, invoked with arguments `foo` and `bar`, and the script contains the following lines:

```bash
  echo $2
  oldargs=$*
  set tarzan jane
  echo $2
  set $oldargs
  echo $2
```

```bash
  outputs bar
  oldargs is "foo bar"
  outputs jane
  resets original arguments
  outputs bar
```
Debugging

Debugging shell scripts is notoriously difficult. The shell hardly provides for a friendly development environment. However, some features of bash help along the way. To run a script in “debug mode”, you have to invoke the script via an explicit bash invocation. Three command-line options to bash are useful:

- `bash -n scriptfile` checks the script for syntax errors, without in fact executing the script.
- `bash -v scriptfile` executes the script, outputting the script line being executed before it is executed.
- `bash -x scriptfile` executes the script, outputting the script line after all shell expansion has been performed being executed.