POSIX Shell Script

Variables

Set a value to environment variable
VARIABLE=Value
using variable
echo $SHELL

Comments

# this is a comment

test *condition* or [*condition*]

test is a program for strings, numerical and file checks. test *expression* can be expressed [*expression*] witch tends to be how its written in scripts. Some common condition.

String test

-n **str**	check if string is not empty
-z **str**	check if string is empty
str = **str**	String equal
str != **str**	String inequality

Numerical test

num -eq **num**	check if numbers are equal
num -ne **num**	check if numbers are unequal
num -lt **num**	check if less than
num -le **num**	check if less or equal than

File test

-f **file**	check if it is file
-d **file**	check if it is a directory
-r **file**	check if file is readable
-w **file**	check if file is writable
-x **file**	check if file is executable

for more condition: **man test**

Combinding statments

stmt1 **&&** *stmt2* # run stmt2 if stmt1 was true
stmt1 || *stmt2* # run stmt2 if stmt1 was false
\ at the end of a line continues it on the next

Command Substitution

echo $(ls)

String handling

'text' all text will be interpret as it
"text" can use shell injections (using variables or commands)

if stmt

if *stmt* **then** *stmts*
elif *stmt* **then** *stmts* # optional
else *stmts* # optional
fi

while stmt

while *stmt* **do** *stmts* **done**
break # exits loop
continue # starts next iteration of loop

until stmt

until *stmt* **do** *stmts* **done**

for stmt

for *x* **in** *values* **do** *stmts* **done**
For each iteration of the loop x will get a new string value from the pool in *values.*

switch case stmt

case var **in**
pattern) *stmts* **;;** # can be several
***) *stmts* **;;** # default catch
esac
Patterns can be written *pattern* | *pattern* for matching several patterns.

functions

functionName(){*stmts*}
return # breaks function and returns
exit *num* # exit and returns num as exit value

Using parameters inside a function

$0 teling you the function name
$1 using the first parameter
$@ printing all parameters
shift will shift all in parameters to a funciton to be shifted, $1=$2 ....
$# number of passed function arguments
$? exes status of last run program
$! PID of last background job
$$ PID of current shell

Debug

set +x # enable debug prints
set -x # stops debug prints
VARIABLE=something *stmt*
This will set the VARIABLE to something and it can be used inside that stmt however will not be set on the other line.

Running programs in background

stmt &
jobs print ongoing jobs
fg bring job 1 to forground
bg bring current job to background
%x # bring job x to forground **kill -9** %x # sends signal to job x

Directing output

prog1 | *prog2* # *prog1* output will become *prog2* input
stmt > *file* # writes the output of stmt to the new file with filename
stmt >> *file* # appends file with filename with output of stms
stmt 2 > &1 # direct *stderr* to *stdout*
number > will tell what file descriptor, if omitted 1 will be assumed. 1 is stdout, 2 is stderr

Expansion

Arithmetic Expansion

expr
$((*expression*)) # return value of expression
* # power
$(($((10**2)) - 50)) # returns 50

Pattern Matching

matches zero or all characters
? matches zero or one character
+ matches one character