Bash Cheat Sheet

Strings:

Operators

${varname:-word}	If varname exists and isn't null, return its value; otherwise return word.
${varname:=word}	If varname exists and isn't null, return its value; otherwise set it to word and then return its value.
${varname:?message}	If varname exists and isn't null, return its value; otherwise print varname: followed by message, and abort the current command or script.
${varname:+word}	If varname exists and isn't null, return word; otherwise return null.
${varname:offset:length}	Performs substring expansion. It returns the substring of $varname starting at offset and up to length characters. If length is omitted substring expansion starts at offset and continues to end of $varname

Pattern Matching

${variable#pattern}	If the pattern matches the beginning of the variable's value, delete the shortest part that matches and return the rest.
${variable##pattern}	If the pattern matches the beginning of the variable's value, delete the longest part that matches and return the rest.
${variable%pattern}	If the pattern matches the end of the variable's value, delete the shortest part that matches and return the rest.
${variable%%pattern}	If the pattern matches the end of the variable's value, delete the longest part that matches and return the rest.
${variable/pattern/string}	The longest match to pattern in variable is replaced by string.
${variable//pattern/string}	All matches to pattern in variable are replaced by string

Condition Tests

Example: [condition]

Operator	True if
string1 = string 2	string1 matches string2
string1 != string 2	string1 does not match string2
string1 == string2	string1 is equal to string2
string1 !== string2	string1 is not equal to string2
string1 < string2	string1 is less than string2
string1 > string2	string1 is greater than string2
-n string1	string1 is not null
-z string1	string1 is null
&&	Logical AND
||	Logical OR

File Condition Tests

Example: [condition]

Operator	True If
-a file	file exists
-d file	file exists and is a directory
-f file	file exists and is a regular file (e.g. is not a directory)
-r file	You have read permission on file. Can also be used with -w, -x for write, and execute permissions respectively.
-s file	file exists and is not empty
file1 -nt file2	file1 is newer than file2
file1 -ot file2	file1 is older than file2

Integers

Setting Variables

declare can be used with options to set variables. For example: `declare -i var1`

-a	The variables are treated as arrays
-i	The variables are treated as integers
-r	Makes the variable read only

Operators

Use with double parenthesis. For example: `echo $((var1++))`

Operator	**Meaning**
++	Increment by one (prefix and postfix)
--	Decrement by one (prefix and postfix)
+, -, *, /	Add, subtract, multiply, and divide respectively
%	Remainder of division
**	Exponentiation

Conditionals

Integer variables take different conditionals than strings. For example: `[ 3 -gt 2 ]`

Operator	**Meaning**
-lt	Less than
-gt	Greater than
-le	Less than or equal to
-ge	Greater than or equal to
-eq	Equal to
-ne	Not equal to

Alternately, the regular operators can be called out, provided the expression is surrounded by double parenthesis: eg: `echo $(( (3 > 2) && (4 <= 1) ))`

Arrays

Storing

array1[2]=value	Will store value as the second element of array1
array1=([2]=value [0]=..)	Will store value as the second element of array1...
array1=(value value)	Will store values in array1 in the order they are entered

Recalling

${array1[0]}	will return element 0 of array1
${array1[*]}	will return all elements of array1
${!array1[*]}	will return occupied array1
${#array1[1]}	will return the length of element 1
${#array1[*]}	will return the length of the array1

Loops

If

```
if condition
then
statements
[elif condition
then
statements]
[else
statements]
fi
```

For

```
for name [in list]
do
statements that can use
$name
done
```

Case

```
case expression in
pattern1  )
        statements ;;
pattern2  )
        statements ;;
esac
```

Select

```
select name [in list]
do
        statements that
        can use $name
done
```

While

```
while condition
do
statements
done
```