Script Languages Quick Reference for Tests Linux Bash

Bahs Shell e Ferramentas

grep prints lines that contain a match for a pattern. Options

-e pattern	Use pattern as the pattern.
-f file	Obtain patterns from file, one per line.
-I	Ignore case distinctions, so that characters that differ only in case match each other.
-v	Invert the sense of matching, to select non-matching lines.
-E	extended regular expressions

Sort merge, or compare all the lines from the files given (or standard input.) Options

-n Sort numerically:

-t SEPARATOR Use character SEPARATOR as the field separator when finding the sort keys in each line. By default, fields are separated by the empty string between a non-whitespace and a whitespace character.

-k POS1[,POS2] The recommended, POSIX, option for specifying a sort field.

cut Print selected parts of lines from each FILE to standard output.

Options : -d DELIM use DELIM instead of TAB for field delimiter -f LIST select only these fields;

wc - print newline, word, and byte counts for each file. With no FILE, read standard input. Options -c, --chars print the character counts -l, --lines print the newline counts -w, --words print the word counts

BASH IF Syntax:	if [expression]; then <commands;> else <commands;> fi
BASH FOR Syntax	for variable in [List_OF_Values] do <commands;> done
Bash While Syntax	while [expression]; do <commands>; done

Regular Expressions

abc…	Letters	[0-9]	Numbers 0 to 9	\s	Any Whitespace
123…	Digits	\w	Any Alphanumeric character	\S	Any Non-whitespace character
\d	Any Digit	\W	Any Non-alphanumeric character		
\D	Any Non-digit character			^…$	Starts and ends
.	Any Character	{m}	m Repetitions	(…)	Capture Group
\.	Period	{m,n}	m to n Repetitions	(a(bc))	Capture Sub-group
[abc]	Only a, b, or c	*	Zero or more repetitions	(.*)	Capture all
[^abc]	Not a, b, nor c	+	One or more repetitions	(abc|def)	Matches abc or d
[a-z]	Characters a to z	?	Optional character		

AWK

<u>AWK Built in Variables</u>

FS	Field separator (default=whitespace)
RS	Record separator (default=\n)
length	The length of the current Record
NF	Number of fields in current record
NR	Number of the current record
OFS	Output field separator (default=space)
ORS	Output record separator (default=\n)
FILENAME	Current filename

<u>Printing</u>

print	Print a record, or field, or value
printf	Printing syntax similar to ISO C

printf format_string, item1, item2, …

<u>Awk conditional statements and loops</u>

if (conditional-expression) action

if (…) action1 else action2

while (condition) action

do action while (condition)

for (initialization; condition; in(de)crement) action

break:	Jump out of enclosing loop
continue:	Skip over the rest of the loop
exit:	Stop executing the script and exit

Script Languages Quick Reference for Tests JavaScript (Mozilla Reference)

The JavaScript Math Object.

- The sqrt() method returns the square root of a number.
- The random() method returns a random number from 0 (inclusive) up to but not including 1 (exclusive).
- The round() method rounds a number to the nearest integer.
- The myrandomString() method returns a short random string.

JavaScript Output

- `console.log(`*`obj1`*` [, `*`obj2`*`, ..., `*`objN`*`])` OR `console.log(`*`msg`*` [, `*`subst1`*`, ..., `*`substN`*`])`
  `obj1` ... `objN` : A list of JavaScript objects to output. The string representations of each of these objects are appended together in the order listed and output
  `Msg` : A JavaScript string containing zero or more substitution strings. `subst1` ... `substN` `are` JavaScript objects with which to replace substitution strings within `msg`.

For loops

- The **for statement** creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement (usually a block statement) to be executed in the loop.
  `for ([`*`initialization`*`]; [`*`condition`*`]; [`*`final-expression`*`])` *`statement`*
- The **for/in** statement loops through the properties of an object. The block of code inside the loop will be executed once for each property. `for (`*`variable`*` in `*`object`*`)` `statement`

The JavaScript Array Object. Properties and Methods

- The **`length`** property of an object which is an instance of type `Array` sets or returns the number of elements in that array. The value is an unsigned, 32-bit integer that is numerically greater than the highest index in the array.
- Array.prototype.**sort()** : The sort() method sorts the elements of an array in place and returns the array. Syntax arr.sort(*compareFunction*). Returns the sorted array. Note that the array is sorted in place, no copy is made. Parameters *compareFunction* : Specifies a function that defines the sort order. compareFunction(a,b) must return zero if a==b, -1 if a<b and 1 if a>b
- Array.prototype.**reduce**(): The reduce() method reduces the array to a single value. The reduce() method executes a provided function for each value of the array (from left-to-right). The return value of the function is stored in an accumulator (result/total). Sytnax array.reduce(*function(total, currentValue, currentIndex, arr), initialValue*). Where currentindex,arr amd initialvalue are optional
- Array.prototype.**pop()** : removes the last element from an array and returns that element. This method changes the length of the array.
- Array.prototype.**push**() : adds one or more elements to the end of an array, returns the new length of the array.
- Array.prototype.**shift**(): removes the first element from an array and returns that element. This method changes the length of the array.
- Array.prototype.**unshift**(): adds one or more elements to the beginning of array, returns the new length of the array.

Lambda/Arrow functions These function expressions are best suited for non-method functions, and they cannot be used as constructors. Basic Syntax

(param1, param2, …, paramN) => { statements }

(param1, param2, …, paramN) => expression // equivalent to: (param1, param2, …, paramN) => { return expression; }

// Parentheses are optional when there's only one parameter name:

typeof *`Syntax:   typeof operand   or   typeof (operand)`*

The typeof operator is used to get the data type (returns a string) of its operand. There are six possible values that typeof returns: "object" "boolean" "function" "number" "string" "undefined"

`JSON.stringify()` converts a JavaScript object or value to a JSON string, optionally replacing values if a replacer function is specified or optionally including only the specified properties if a replacer array is specified.