Bash cheat sheet

1 Variable handling

1.1 General stuff

{`$#varname`}	Expands into the length of `varname` (number of characters).
`$(UNIX command)`	Expands to the output to stdout by `UNIX command`. Can be nested (example: `$(ls $(pwd))`. `$(< "$filename")` is the contents of the file `"$filename"` with any trailing newlines removed.

1.2 Script arguments

`$0`	Expands into the name of the script/function called. If function is called within script it will still hold the name of the script
`$1, $2, $3... $N`	Expands into each argument sent to script/function.
`"$*"`	Expands into a single string containing all the arguments recieved by the script/function (except `$0`) seperated by the value of the environmental variable IFS.
`"$@"`	Expands into `"$1" "$2" "$3"... "$N"`.
`$#`	Expands into number of arguments (not counting `$0`).

1.3 String operators

`${varname:-word}`	If `varname` exists and isnt null, return its value. Otherwise return `word`.
`${varname:=word}`	If `varname` exists and isnt null, return its value. Otherwise set `varname`'s value to `word` and return that value (positional and special parameters cannot be assigned this way).
`${varname:?message}`	If `varname` exists and isnt null, return its value. Otherwise print `"varname:"` followed by message.
`${varname:+word}`	If `varname` exists and isnt null, return `word`. Otherwise return null.
`${varname:offset:length}`	Returns the substring of `$varname` starting at `offset` and up to `length` characters. The first character has position 0. If `length` is omitted the substring starts at `offset` and continues to the end of `$varname`. If `offset` is less than 0 then the position is taken from the end of `$varname`. If varname is `"@"`,`length` is the number of positional parameters starting at `offset`.
`${varname:-word}`	`"$string1"` is null.

1.4 Patterns and pattern matching

`${varname#pattern}`	If `pattern` matches the beginning of `varname`'s value, delete the shortest part that matches and return the result.
`${varname##pattern}`	If `pattern` matches the beginning of `varname`'s value, delete the longest part that matches and return the result.
`${varname%pattern}`	If `pattern` matches the end of `varname`'s value, delete the shortest part that matches and return the result.
`${varname%%pattern}`	If `pattern` matches the end of `varname`'s value, delete the longest part that matches and return the result.
`${varnamepatternstring}`	The first match of `pattern` in `varname`'s value is replaced by `string`. If `pattern` begins with a #, it must match the start of `varname`. If it begins with a %, it must match the end of `varname`. If `string` is null, the match is deleted. If `varname` iss or , the operation is applied to each positional parameter in turn and the expansion is the resultant list.
`${varnamepatternstring}`	All matches of `pattern` in `varname`'s value is replaced by `string`. If `pattern` begins with a #, it must match the start of `varname`. If it begins with a %, it must match the end of `varname`. If `string` is null, the matches are deleted. If `varname` iss or , the operation is applied to each positional parameter in turn and the expansion is the resultant list.

2 Conditionals

2.1 General stuff

2.1.1 if/else

```
if command
then
...
fi
```

or

```
if [condition]
then
...
fi
```

Example:

```
if command
then
...
elif [condition]
then
...
else
...
fi
```

2.1.2 for

```
for variable in list
do
...
done
```

Example 1:

```
for variable in 1 2 3 4 5
do
echo "Iteration $variable"
done
```

Output:

```
./my_script
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

Example 2:

```
for variable in $1 $2 $3
do
echo "Arg:  $variable"
done
```

Output:

```
./my_script one two three
Arg:  one
Arg:  two
Arg:  three
```

Example 3:

```
for variable in "$@"
do
echo "Arg:  $variable"
done
```

Output:

```
./my_script one two three
Arg:  one
Arg:  two
Arg:  three
```

Example 4:

```
FSH=:  for variable in "$PATH"
do
echo "$variable"
done
```

Output:

```
./my_script
/usr/bin
/bin
/sbin
/usr/local/bin
```

2.2 String comparison

`[ "$string1" = "$string2" ]`	`"$string1"` matches `"$string2"`.
`[ "$string1" != "$string2" ]`	`"$string1"` does not match `"$string2"`.
`[ "$string1" < "$string2" ]`	`string` 1 is less than `"$string2"` (strcmp).
`[ "$string1" > "$string2" ]`	`"$string1"` is greater than `"$string2"`.
`[ -n "$string1" ]`	`"$string1"` is not null.
`[ -z "$string1" ]`	`"$string1"` is null.

2.3 File attribute checking

`[ -a "$filename" ]`	`"$filename"` exists.
`[ -d "$filename" ]`	`"$filename"` exists and is a directory.
`[ -e "$filename" ]`	`"$filename"` exists (same as`-a`).
`[ -f "$filename" ]`	`"$filename"` ”$filename” exists and is a regular file (not a directory or special type of file).
`[ -r "$filename" ]`	You have read permission on `"$filename"`.
`[ -s "$filename" ]`	`"$filename"` exists and is not empty.
`[ -w "$filename" ]`	You have write permission on `"$filename"`.
`[ -x "$filename" ]`	You have execute permission on `"$filename"`, or directory search permission if it is a directory.
`[ -N "$filename" ]`	`"$filename"` was modified since it was last read.
`[ -O "$filename" ]`	You are the owner of `"$filename"`.
`[ -G "$filename" ]`	`"$filename"`’s group ID matches yours.
`[ "$filename1" -nt "$filename2" ]`	`"$filename1"` is newer than `"$filename2"`.
`[ "$filename1" -ot "$filename2" ]`	`"$filename1"` is older than `"$filename2"`.

2.4 Integer conditionals

`[ "$varname1" -lt "$varname2" ]`	`"$varname1"` is lesser than `"$varname2"`.
`[ "$varname1" -le "$varname2" ]`	`"$varname1"` is lesser than or equal to `"$varname2"`.
`[ "$varname1" -gt "$varname2" ]`	`"$varname1"` is greater than `"$varname2"`.
`[ "$varname1" -ge "$varname2" ]`	`"$varname1"` is greater than or equal to `"$varname2"`.
`[ "$varname1" -eq "$varname2" ]`	`"$varname1"` is equal to `"$varname2"`.
`[ "$varname1" -ne "$varname2" ]`	`"$varname1"` is not equal to `"$varname2"`.