Expansions and substitutions

Before executing your commands, Bash checks whether there are any syntax elements in the command line that should be interpreted rather than taken literally. After splitting the command line into tokens (words), Bash scans for these special elements and interprets them, resulting in a changed command line: the elements are said to be **expanded** to or **substituted** to **new text and maybe new tokens** (words).

The most simple example of this behaviour is a referenced variable:

```
mystring="Hello world"
echo "$mystring"
```

The `echo` program definitely doesn't care about what a shell variable is. It is Bash's job to deal with the variable. Bash **expands** the string " `$mystring` " to " `Hello world` ", so that `echo` will only see `Hello world` , not the variable or anything else!

After all these expansions and substitutions are done, all quotes that are not meant literally (i.e., the quotes that marked contiguous words, as part of the shell syntax) are removed from the commandline text, so the called program won't see them. This step is called **quote-removal**.

Overview

Saw a possible expansion syntax but don't know what it is? Here's a small list.

- Parameter expansion (it has its own overview section)
  - `$WORD`
  - `${STUFF…}`
- Pathname expansion
  - `*.txt`
  - `page_1?.html`
- Arithmetic expansion
  - `$(( EXPRESSION ))`
  - `$[ EXPRESSION ]`
- Command substitution
  - `$( COMMAND )`
  - `` ` COMMAND ` ``
- Tilde expansion
  - `~`
  - `~+`
  - `~-`
- Brace expansion

    - `{X,Y,Z}`
    - `{X..Y}`
    - `{X..Y..Z}`
- Process substitution
    - `<( COMMAND )`
    - `>( COMMAND )`

Order

Bash performs expansions and substitutions in a defined order. This explains why globbing (pathname expansion), for example, is safe to use on filenames with spaces (because it happens **after** the final word splitting!).

The order is (from first to last):

- Brace expansion
- Tilde expansion
- The following expansions happen at the same time, in a left-to-right fashion on the commandline (see below)
    - Parameter expansion
    - Arithmetic expansion
    - Command substitution
- Word splitting
- Pathname expansion

Process substitution is performed **simultaneously** with parameter expansion, command substitution and arithmetic expansion. It is only performed when the underlying operating system supports it.

The 3 steps parameter expansion, arithmetic expansion and command substitution happen at the same time in a left-to-right fashion on nthe commandline. This means

```
i=1
echo $i $((i++)) $i
```

will output `1 1 2` and not `1 1 1`.

Discussion