Accessing bash command line args $@ vs $*

In many SO questions and bash tutorials I see that I can access command line args in bash scripts in two ways:

```
$ ~ >cat testargs.sh
#!/bin/bash

echo "you passed me" $*
echo "you passed me" $@
```

Which results in:

```
$ ~> bash testargs.sh arg1 arg2
you passed me arg1 arg2
you passed me arg1 arg2
```

What is the difference between `$*` and `$@` ?
When should one use the former and when shall one use the latter?

bash command-line-arguments

edited Nov 3 '16 at 7:09
Seanny123
1,360 2 18 45

asked Sep 7 '12 at 8:28
Oz123
9,961 12 52 104

take a look at this answer: stackoverflow.com/a/842325/671366 – codeling Sep 7 '12 at 8:32

1 @nyarlathotep, thanks, the question is useful. The title is awful. I would never guess the answer shows what I was looking for. Somehow, my question is indeed trivial, but the way I formulated it in my head, was not easy to find in the bash manual (although /* brings you to the right place ...) – Oz123 Sep 7 '12 at 8:48

4 Answers

The difference appears when the special parameters are quoted. Let me illustrate the differences:

```
$ set -- "arg  1" "arg  2" "arg  3"

$ for word in $*; do echo "$word"; done
arg
1
arg
2
arg
3

$ for word in $@; do echo "$word"; done
arg
1
```

```
arg
2
arg
3

$ for word in "$*"; do echo "$word"; done
arg  1 arg  2 arg  3

$ for word in "$@"; do echo "$word"; done
arg  1
arg  2
arg  3
```

one further example on the importance of quoting: note there are 2 spaces between "arg" and the number, but if I fail to quote $word:

```
$ for word in "$@"; do echo $word; done
arg 1
arg 2
arg 3
```

and in bash, `"$@"` is the "default" list to iterate over:

```
$ for word; do echo "$word"; done
arg  1
arg  2
arg  3
```

edited Feb 22 '14 at 11:20

answered Sep 7 '12 at 10:46
glenn jackman
132k 20 107 185

24 +1 I've always thought this concept was best demonstrated by a simple example, in which the bash manual is completely lacking. – chepner Sep 7 '12 at 12:01

5 Is there a possible use case, when `$*` or `"$*"` may be required, & the purpose cannot be served by `$@` or `"$@"` ? – anishsane Aug 3 '13 at 11:17

1 yes, exactly as demonstrated here: only when quoted. – glenn jackman Aug 3 '13 at 11:29

2 Which version is more suitable for a "wrapper" script, where the scripts parameters need to become parameters to a new command? – Segfault Mar 30 '15 at 15:53

4 @Segfault, in this case, always choose `"$@"` with the quotes. – glenn jackman Mar 30 '15 at 16:09

A nice handy overview table from the Bash Hackers Wiki:

Syntax	Effective result
`$*`	`$1 $2 $3 … ${N}`
`$@`	`$1 $2 $3 … ${N}`
`"$*"`	`"$1c$2c$3c…c${N}"`
`"$@"`	`"$1" "$2" "$3" … "${N}"`

answered Jan 22 '15 at 22:18
Serge Stroobandt
5,535 3 38 38

19 ... where "c" is the first character of $IFS – glenn jackman Dec 22 '15 at 20:11

16 ...and `$IFS` stands for "Internal Field Separator." – Serge Stroobandt Dec 25 '15 at 18:43

$*

> Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, it expands to a single word with the value of each parameter separated by the first character of the IFS special variable. That is, "$*" is equivalent to "$1c$2c...", where c is the first character of the value of the IFS variable. If IFS is unset, the parameters

> are separated by spaces. If IFS is null, the parameters are joined without intervening separators.

$@

> Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, each parameter expands to a separate word. That is, "$@" is equivalent to "$1" "$2" ... If the double-quoted expansion occurs within a word, the expansion of the first parameter is joined with the beginning part of the original word, and the expansion of the last parameter is joined with the last part of the original word. When there are no positional parameters, "$@" and $@ expand to nothing (i.e., they are removed).

Source: Bash man

answered Sep 7 '12 at 8:36
Muffo
974 10 21

$@ is same as $*, but each parameter is a quoted string, that is, the parameters are passed on intact, without interpretation or expansion. This means, among other things, that each parameter in the argument list is seen as a separate word.

Of course, "$@" should be quoted.

http://tldp.org/LDP/abs/html/internalvariables.html#ARGLIST

answered Sep 7 '12 at 8:32
rkosegi
6,514 3 29 56
