How to make your Perl script (or other) Bash aware

19.02.25

Pegasi Knowledge
https://ghost.pegasi.fi/wiki/

Table of Contents

How to make your Perl script (or other) Bash aware

A short explanation on how to make your script support Bash completion feature when hitting tab key AND give your script total control of the choices provided. I use Perl example but it is very doable with any other language as well as it is a Bash feature mostly.

In practice Bash completion happens when you do following:

```
myscript <tab>
choice1  choice2 choice3
```

You get choices in argument number one when you hit tab key twice.

```
myscript xxx<tab>
xxxchoice1  xxxchoice2
```

You get choices beginning with xxx when you hit tab key twice or you get a complete argument if number of choices equals one.

Bash completion configuration

You need to create a file /etc/bash_competion.d/myscript.bash with following contents

```
_myscript ()
{
        COMPREPLY=()
        cur="${COMP_WORDS[COMP_CWORD]}"
        prev="${COMP_WORDS[COMP_CWORD-1]}"

        local choices=$(
                for x in `COMP_CWORD=$COMP_CWORD perl /path/to/myscript
complete ${COMP_WORDS[@]}`
                do echo -n "${x} ";
                done)
        COMPREPLY=( $(compgen -W "${choices}" -- ${cur}) )
        return 0
}
complete -F _myscript myscript
```

Please note the parameter **complete** for the script. This is to signal to your script that we are asking completion choices and this functionality we will do below. You can rename it to whatever you want.

Script Bash completion support

We need to support argument **complete** (or whatever you want to name it) in our script and we can do it with Perl in following manner. First we recognize the argument:

```
if ($ARGV[0] =~ /^complete$/) {
        complete_command(@ARGV);
        exit 0;
}
```

Then we make the subroutine to handle the request. Look explanations below.

```
sub complete_command {
        my @arg = @_;
        my $index = $ENV{COMP_CWORD};
        $index--;
        my $curr = $arg[$index];
        my $prev = $arg[$index-1];
        if ($index == 0) {
                if ($curr =~ /^[^\-]{2}/) {
                        @matches = get_args($curr);
                } elsif ($curr =~ /^$/ || $curr =~ /^-$/ || $curr =~ /^-.$/)
{
                        @matches = ("-a", "-b", "-c");
                }
        } elsif ($prev =~ /^-[a]/) {
                @matches = get_option_a($curr);
        } elsif ($prev =~ /^-[b]/) {
                @matches = get_option_a($curr);
        } elsif ($prev =~ /^-[c]/) {
                @matches = get_option_a($curr);
        } else {
                @matches = get_args($curr);
        }
        print join(" ", @matches);
}
```

First we save the arguments in an array and we get the current argument number from Bash completion so that we will not process anything we've already done. Then we save current and previous arguments into their own variables for clarity's sake. Then we process the first argument and following arguments and recognize options with leading '-' characters.

Outcome

Eventually we return what Bash completion wants: a space delimited list of choices. Also separate lines may work but spaces seem to be more common.

This can and should be refined so that the first arguments affect the rest of the choices meaning for example if you do come contexting in your first arguments you will get fewer choices following that.