Is it possible to specify the order getopts conditions are executed?

Asked 11 years, 10 months ago Modified 11 years, 10 months ago Viewed 8k times

3

In a bash script, I'd like to load settings from a config file and override individual settings with command-line options. In cases where a setting is specified both in the config file and also on the command line, the command-line setting should take precedence.

How do you ensure the config file is loaded before the other getopts blocks? Here's what I've got:

```
#!/bin/bash
# ...

while getopts “c:l:o:b:dehruwx” OPTION
do
   case $OPTION in
      c)
         echo "load"
         CONFIG_FILE=$OPTARG
         # load_config is a function that sources the config file
         load_config $CONFIG_FILE
         ;;
      l)
         echo "set local"
         LOCAL_WAR_FILE=$OPTARG
         ;;

# ...

   esac
done
shift $(($OPTIND - 1))
```

No matter what order I put the handler for the -c option, it always loads the config file AFTER the other options are set. This makes it more of a pain to merge the config file settings with the command-line options.

bash shell

Share Improve this question Follow

asked Feb 26, 2013 at 0:09

Travis Bear

13.8k ● 9 ● 44 ● 52

What does `$CONFIG_FILE` do exactly? – Explosion Pills Feb 26, 2013 at 0:23

Why not just run getopts twice, once for "c:", and again for "c:l:o:..." – Brad Lanam Feb 26, 2013 at 0:29

@Brad Lanam I could go this way. It doesn't feel ideal, but I imagine it would work.
– Travis Bear Feb 26, 2013 at 1:11

2 Answers

Sorted by: Highest score (default)

11

Each call to `getopts` always processes the "next" option (as determined by examining `$OPTIND`), so your `while`-loop will necessarily process the options in the order they appear.

Since you want `-c` to be partly superseded by other options, even if it appears after them on the command-line, there are a few approaches you can take.

One is to loop over the options *twice*:

```
#!/bin/bash
# ...

optstring='c:l:o:b:dehruwx'

while getopts "$optstring" OPTION
do
   case $OPTION in
      c)
         echo "load"
         CONFIG_FILE=$OPTARG
         # load_config is a function that sources the config file
         load_config $CONFIG_FILE
   esac
done

OPTIND=1

while getopts "$optstring" OPTION
do
   case $OPTION in
      l)
         echo "set local"
         LOCAL_WAR_FILE=$OPTARG
         ;;
      # ...
   esac
done
shift $(($OPTIND - 1))
```

Another is to save options in variables that `-c` *won't* override, and then copy them over:

```
#!/bin/bash
# ...

while getopts c:l:o:b:dehruwx OPTION
do
   case $OPTION in
      c)
         echo "load"
         CONFIG_FILE=$OPTARG
         # load_config is a function that sources the config file
         load_config $CONFIG_FILE
         ;;
      l)
         echo "set local"
         LOCAL_WAR_FILE_OVERRIDE=$OPTARG
         ;;
      # ...
   esac
done
shift $(($OPTIND - 1))

LOCAL_WAR_FILE="${LOCAL_WAR_FILE_OVERRIDE-${LOCAL_WAR_FILE}}"
```

(Or, conversely, the config file can set options like `LOCAL_WAR_FILE_DEFAULT`, and then you'd write `LOCAL_WAR_FILE="${LOCAL_WAR_FILE-${LOCAL_WAR_FILE_DEFAULT}}"`.)

Another option is to require that `-c`, if present, come *first*. You can do that by handling it first yourself:

```
if [[ "$1" = -c ]] ; then
   echo "load"
   CONFIG_FILE="$2"
   # load_config is a function that sources the config file
   load_config "$CONFIG_FILE"
   shift 2
fi
```

and then in your main `while`-loop, just handle `-c` by printing an error message.

Another is simply to document your existing behavior and call it a "feature". A lot of Unix utilities have later options supersede earlier ones, so this behavior isn't really a problem.

Share

Improve this answer

Follow

edited Feb 26, 2013 at 22:07

answered Feb 26, 2013 at 18:20

ruakh

183k ● 29 ● 289 ● 321

In the first example, what's the purpose of OPTIND? – liv913 Nov 30, 2017 at 13:32

1 @liv913: OPTIND is how getopts keeps track of which argument it's processing. Resetting it to 1 tells it to jump back to the first argument. – ruakh Nov 30, 2017 at 16:26

Can you better explain the second example? I don't see any difference. In the third example you expect `-c` to be the 1st argument, but what happens with `call.sh -l -c` ?
– João Pimentel Ferreira Dec 9, 2017 at 19:52

can you explian what `OPTIND=1` makes on the 1st example? I realised it makes the trick, since without it, the next `while` loop is not run. – João Pimentel Ferreira Dec 9, 2017 at 20:11

At the end of the first example, why do you call `shift $(($OPTIND - 1))` ? Does that just place the cursor at the spot after where ever that second option is? – Brian Oct 3, 2018 at 0:41

1

Assuming your config file contains the default options for your program, you should always use those options by default unless they are overridden by their equivalent command-line options. This is reasonable. In your case, simply source/load the config file first and then parse command-line options - and assigning new values to them in the `parseopts` loop as needed.

Share Improve this answer Follow

answered Feb 26, 2013 at 0:29

user1019830

This is exactly what I want to do. The question is, how do do do this when the config file is itself one of the options being parsed? – Travis Bear Feb 26, 2013 at 2:35