

# Handling of shell builtins in make(1)

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2005-05/0076.html>

---

**From:** Harti Brandt ([hartmut.brandt\\_at\\_dlr.de](mailto:hartmut.brandt_at_dlr.de))

**Date:** 05/23/05

Date: Mon, 23 May 2005 15:51:16 +0200 (CEST)

To: arch@freebsd.org

Hi all,

I think I found a problem in the shell code in make(1), but I'm not sure whether to fix it or not and how. The problem is as follows: in compat mode (this is the default mode when make(1) is not called with -j) the command lines of a target are executed by one shell per line (this is also how Posix wants it). To reduce the number of shells make does an optimisation: when the command line does not contain one of a pre-defined set of meta characters and does not start with one of a predefined set of shell builtins, make directly exec's the command instead of using an intermediate shell. The problem is that the current list of builtins is limited to:

```
alias cd eval exec exit read set ulimit unalias umask unset wait
```

Obviously this is not the full set of shell builtins and does also not contain the shell reserved words.

The result of this is that for one and the same command you can get different behaviour whether you execute it via make(1) or via sh -c '...'. As an example take echo(1). When called via the sh -c you get the builtin echo which supports the -e option, when executed by make(1) you get /bin/echo which doesn't. If you suddenly include a shell meta character in the command line:

```
foo:
    echo "MAKEFLAGS: ${MAKEFLAGS}"
```

you suddenly get also the builtin (':' is a meta character).

For the reserved words the situation is almost the same. With the following makefile:

```
foo:
    if
```

## freebsd-arch: Handling of shell builtins in make(1)

one gets:

```
if:No such file or directory, while
```

```
foo:
```

```
    if [ -x /bin/sh ] ; then echo hey ; fi
```

you get what you expect.

I think all this may be very confusing. The question is what to do. I see the following options:

1. leave it as it is.
2. include the Posix reserved words and builtins into the list.
3. include all reserved words and builtins of our shell into the list.

Option (3) is probably best. With (2) you still get different behaviour for the two command lines in:

```
foo:
```

```
    bind -h
```

```
    bind -h *
```

(the first line will try to find bind in the path while the second will execute the shell builtin).

Opinions?

harti

---

freebsd-arch@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"